

USER GUIDE

Essential Studio

for EJ2 TypeScript

Version - v24.1.41 | Release Date - December 18, 2023

RadioButton	53
Label and size in EJ2 JavaScript Radio button control	53
Label	53
Size	54
See Also	55
Accessibility in EJ2 JavaScript Radio button control	55
WAI-ARIA attributes	56
Keyboard interaction	57
Ensuring accessibility	57
See also	57
How To	57
Customize radiobutton appearance in EJ2 JavaScript Radio button control	57
Name and value in form submit in EJ2 JavaScript Radio button control	60
Right to left in EJ2 JavaScript Radio button control	62
Set the disabled state in EJ2 JavaScript Radio button control	63
Ej1 api migration in EJ2 JavaScript Radio button control	65
Properties	65
Methods	66
Events	66
Range Navigator	67
Getting started in EJ2 JavaScript Range navigator control	67
Dependencies	67
Setup for Local Development	67
Configuring System JS	67
Add Range Navigator to the Project	68
Run the Application	69
Module Injection	70
Populate Range Navigator with Data	70
Enable Tooltip	71
Selecting range in EJ2 JavaScript Range navigator control	72
Lightweight in EJ2 JavaScript Range navigator control	73
See Also	74
Series types in EJ2 JavaScript Range navigator control	74
Line	74
Area	75

StepLine.....	76
Data in EJ2 JavaScript Range navigator control	77
Numeric.....	77
Logarithmic Axis	82
Date-time	85
Period selector in EJ2 JavaScript Range navigator control	88
Periods	88
Positioning period selector	89
Height.....	90
Visibility of range navigator	91
See Also	92
Labels in EJ2 JavaScript Range navigator control.....	92
Multilevel labels	92
Grouping	93
Smart labels.....	94
Label positioning	95
Labels customization.....	96
Grid tick in EJ2 JavaScript Range navigator control	97
Grid line customization	97
Tick line customization.....	98
Customization in EJ2 JavaScript Range navigator control	99
Navigator appearance.....	99
Thumb	100
Border customization.....	101
Deferred update.....	102
Allow snapping.....	104
Animation.....	105
See Also	106
Tool tip in EJ2 JavaScript Range navigator control	106
Customization	106
Label Format	107
R t l in EJ2 JavaScript Range navigator control	108
Export print in EJ2 JavaScript Range navigator control	109
Export.....	109
Print.....	110

Accessibility in EJ2 JavaScript Range navigator control	112
WAI-ARIA attributes	113
Keyboard interaction	113
Ensuring accessibility	113
See also	113
Ej1 api migration in EJ2 JavaScript Range navigator control	113
RangeNavigator.....	113
Series.....	118
StyleSettings.....	119
Tooltip	121
Period Selector.....	122
Methods.....	122
Events.....	122
Range Slider	125
Getting started in EJ2 JavaScript Range slider control.....	125
control Initialization	125
See Also	128
Ticks in EJ2 JavaScript Range slider control	129
Step	130
Min and Max	132
Format in EJ2 JavaScript Range slider control	133
Using format API	135
Using Events.....	136
Limits in EJ2 JavaScript Range slider control.....	138
Default and MinRange Slider limits	139
Range Slider limits.....	140
Handle lock.....	142
Globalization and localization in EJ2 JavaScript Range slider control.....	144
Localization	144
See Also	146
Style in EJ2 JavaScript Range slider control	146
Customizing the slider track.....	146
Customizing the slider handle.....	146
Customizing the slider limits	146
Customizing the slider ticks	147

Customizing the slider buttons	147
Accessibility in EJ2 JavaScript Range Slider component	147
WAI-ARIA attributes	148
Keyboard interaction	148
Ensuring accessibility	149
See also	149
Ej1 api migration in EJ2 JavaScript Range slider control	149
How To	151
Format value using slider in EJ2 JavaScript Range slider control	151
Customize the bar in EJ2 JavaScript Range slider control	157
Customize the ticks in EJ2 JavaScript Range slider control	161
ticks_slider.e-scale :nth-child(1)::before {	162
Customize the limits in EJ2 JavaScript Range slider control	166
Customize the thumb in EJ2 JavaScript Range slider control	170
square_slider.e-control.e-slider .e-handle {	171
circle_slider.e-control.e-slider .e-handle {	171
oval_slider.e-control.e-slider .e-handle {	171
Validate slider using formvalidator in EJ2 JavaScript Range slider control	174
Show slider from hidden state in EJ2 JavaScript Range slider control	181
Reversible Range Slider in EJ2 JavaScript	184
Rating	186
Precision modes in EJ2 JavaScript Rating control	186
Labels in EJ2 JavaScript Rating control	188
Label position	189
Label template	191
Tooltip in EJ2 JavaScript Rating control	192
Tooltip template	194
Tooltip customization	196
Selection in EJ2 JavaScript Rating control	198
Min value	199
Single selection	200
Show or hide reset button	202
Templates in EJ2 JavaScript Rating control	203
Empty (unrated) symbol template	203
Full (rated) symbol template	206

Using Emoji icon as rating symbol	208
Using SVG icon as rating symbol	210
Using PNG image as rating symbol	212
Events in EJ2 JavaScript Rating control	214
beforeItemRender	214
created	214
onItemHover	214
valueChanged.....	215
Appearance in EJ2 JavaScript Rating control	216
Items count	216
Disabled.....	219
Visible	221
Read only.....	224
CssClass	226
Changing icon using CssClass	234
Accessibility in EJ2 JavaScript Rating control	236
Keyboard interaction	236
ARIA attribute	237
Ribbon	237
Tabs and Groups	237
Adding Tabs.....	237
Adding Groups	238
Adding Items	240
Ribbon Items	241
Built-in items.....	242
Custom items	262
Items display Mode.....	263
Enable or disable items	266
Ribbon Layouts.....	267
Classic layout.....	267
Simplified layout	288
Minimized State	294
File Menu	296
Visibility	296
Adding menu items	298

Open submenu on click.....	300
Custom header text	302
Ribbon Backstage.....	304
Adding backstage items	304
Adding footer items	308
Adding separator.....	312
Back button	316
Backstage target.....	319
Template	323
Setting width and height.....	327
Help Pane	331
Tooltip	333
Adding Title	333
Adding Content	335
Adding Icon	336
Customization	338
Ribbon Resizing	341
Defining items allowed size	341
Defining items active size.....	343
Events.....	343
tabSelected	343
tabSelecting.....	344
ribbonCollapsing	346
ribbonExpanding	346
launcherIconClick	347
Button item events	348
CheckBox item events	350
ColorPicker item events	351
ComboBox item events	356
DropDown item events	362
SplitButton item events	367
GroupButton item events	374
FileMenu events.....	377
Backstage view events	384
RichTextEditor	385

Getting started in EJ2 JavaScript Rich text editor control.....	385
Component Initialization.....	385
Module Injection.....	394
Configure the Toolbar	394
Insert images and links.....	396
Retrieve the formatted content.....	398
Retrieve the number of characters in the Rich Text Editor	398
See Also	399
Editor modes in EJ2 JavaScript Rich text editor control	399
HTML editor	399
Markdown editor	401
See Also	403
Toolbar in EJ2 JavaScript Rich text editor control.....	403
Expand Toolbar	404
Multi-row Toolbar	405
Floating Toolbar	407
Toolbar items	408
Custom tool.....	413
Quick inline toolbar.....	418
See Also	421
Inline mode in EJ2 JavaScript Rich text editor control.....	421
Show on select/click.....	421
See Also	422
Paste cleanup in EJ2 JavaScript Rich text editor control.....	422
MS Word to HTML	422
Paste cleanup	423
Prompt dialog.....	423
Paste as plain text	423
Keep format	423
Denied tags	424
Denied attributes	424
Allowed style properties	424
Mention integration in EJ2 JavaScript Rich text editor control	426
See Also	429
Enter key in EJ2 JavaScript Rich text editor control.....	429

Enter key customization.....	429
Shift-Enter key customization	432
Styling in EJ2 JavaScript Rich text editor control	434
Font name and size	434
Custom font and size	436
Font and Background color	438
Editor content styles	440
See Also	445
Image in EJ2 JavaScript Rich text editor control	446
Upload options.....	446
Image delete	448
Insert from web	450
Dimension	451
Caption and Alt Text.....	452
Display position.....	452
Image with link.....	452
Resize	453
Drag and Drop.....	453
See Also	456
Audio in EJ2 JavaScript Rich text editor control.....	456
Configure audio tool in the toolbar	456
Insert audio from the web	458
Insert audio from local machine	458
Replacing audio.....	462
Delete audio	463
Display position.....	464
Rename audio before inserting	464
Upload audio with authentication	467
See Also	468
Video in EJ2 JavaScript Rich text editor control.....	468
Configure the video tool in the toolbar	469
Insert a video from the web.....	470
Insert video from local machine	471
Replacing video	475
Delete video	476

Dimension	477
Display position.....	478
Resize video.....	478
Rename video before inserting.....	479
Upload video with authentication	481
See Also	482
Link in EJ2 JavaScript Rich text editor control.....	483
Insert link	483
Remove Link.....	485
Auto-link.....	485
Manipulation.....	485
See Also	487
Table in EJ2 JavaScript Rich text editor control	487
Insert table	488
Quick Toolbar	489
Table Header.....	490
Insert Rows.....	490
Insert Columns	490
Set Color.....	490
Delete Table	491
Vertical Align	491
Horizontal Align.....	491
Table Styles	491
Table Properties	492
Table cell merge and split	492
Emoji Picker in EJ2 JavaScript RichTextEditor control	495
Enabling the toolbar option and custom emojis.....	495
Using the shortcut key to open the emoji picker.....	499
Navigating and selecting emojis using the keyboard.....	499
Markdown in EJ2 JavaScript Rich text editor control	499
Supported Commands	499
Markdown to HTML	501
Table.....	504
Custom format	512
See Also	514

File browser in EJ2 JavaScript Rich text editor control	514
Required additional package styles and scripts reference	514
Iframe in EJ2 JavaScript Rich text editor control	518
IFrame attributes	520
Adding external CSS/Script File	522
See Also	523
Format Painter in EJ2 JavaScript Rich Text Editor Control Syncfusion.....	524
Enabling the toolbar option for Format Painter	524
Customization of copy and paste format.....	526
Using the shortcut key to copy and paste the format	528
Form support in EJ2 JavaScript Rich text editor control	529
Render the Rich Text Editor	529
Obtain the value.....	529
See Also	531
Validation in EJ2 JavaScript Rich text editor control.....	531
Validation rules	531
Validation message	533
Custom placement of validation message	535
Style in EJ2 JavaScript Rich text editor control	537
Customizing the Rich Text Editor's content	537
Customizing the Rich Text Editor's toolbar	537
Customizing the Rich Text Editor's character count	538
Globalization in EJ2 JavaScript Rich text editor control.....	538
Localization	538
RTL.....	552
Exec command in EJ2 JavaScript Rich text editor control.....	554
Miscellaneous in EJ2 JavaScript Rich text editor control	556
Placeholder	556
Character count	558
Code view.....	560
Undo/Redo Manager	562
Prevention of cross-site scripting (XSS)	564
Resizable support.....	567
Number and Bullet Format Lists	570
Xhtml validation in EJ2 JavaScript Rich text editor control.....	571

Attributes	572
HTML Elements	572
Third party integration in EJ2 JavaScript Rich text editor control	574
Code-Mirror Integration	574
Embed.ly Integration.....	577
Keyboard support in EJ2 JavaScript Rich text editor control	578
HTML formation shortcut key.....	579
Markdown formation shortcut key.....	582
Custom key config.....	584
See Also	587
Accessibility in EJ2 JavaScript Rich text editor control	587
ARIA attributes.....	588
Keyboard interaction	590
Ensuring accessibility	592
See also	592
How To	592
Add google font in EJ2 JavaScript Rich text editor control	592
Default font in EJ2 JavaScript Rich text editor control.....	595
File size in EJ2 JavaScript Rich text editor control.....	597
Shortcut key in EJ2 JavaScript Rich text editor control.....	598
Save in EJ2 JavaScript Rich text editor control.....	600
Placeholder in EJ2 JavaScript Rich text editor control	602
Cursor in EJ2 JavaScript Rich text editor control	603
Rename images in server in EJ2 JavaScript Rich text editor control.....	605
Format code block in EJ2 JavaScript Rich text editor control	608
File attachment in EJ2 JavaScript Rich text editor control.....	610
Schedule.....	613
Getting started in EJ2 JavaScript Schedule control.....	613
Dependencies.....	613
Setup for local environment	613
Adding Syncfusion resources	614
Initialize the Scheduler.....	618
Populating appointments	619
Setting date	621
Setting view.....	621

Individual view customization	621
Module injection in EJ2 JavaScript Schedule control.....	623
Module injection	623
Scheduler interactions in EJ2 JavaScript Schedule control	624
Appointments in EJ2 JavaScript Schedule control	625
Normal events.....	625
Spanned events.....	626
All-day events.....	627
Expand all day appointments view on initial load	627
Customize the rendering of the spanned events.....	630
Recurring events	631
Event fields.....	641
Adding Custom fields	645
Customize the order of the overlapping events	647
Drag and drop appointments.....	649
Inline Appointment	663
Appointment Resizing	666
Appointment customization	672
Setting minimum height	677
Block Dates and Times	678
Readonly	681
Make specific events readonly.....	682
Restricting event creation on specific time slots	686
Differentiate the past time events.....	687
Appointments occupying entire cell	689
How to limit maximum number of events to display	690
Display tooltip for appointments.....	691
Appointment filtering	694
Appointment selection	696
Deleting multiple appointments	697
Retrieve event details from the UI of an event	697
Get the current view appointments	699
Get the entire appointment collections.....	701
Refresh appointments	703
Data binding in EJ2 JavaScript Schedule control.....	703

Binding local data	703
Binding remote data	705
Loading data via AJAX post	710
Passing additional parameters to the server	711
Handling failure actions	712
Scheduler CRUD actions.....	714
Configuring Scheduler with Google API service	718
See Also	720
Crud actions in EJ2 JavaScript Schedule control	720
Add	720
Edit	726
Delete.....	739
Drag and drop	747
Resize	749
Enabling lazy loading for appointments.....	753
See Also	756
Editor template in EJ2 JavaScript Schedule control	757
Event editor.....	757
Customizing event editor using template	769
How to customize header and footer using template	772
Quick popups	787
More events indicator and popup	796
Timezone in EJ2 JavaScript Schedule control	810
Understanding date manipulation in JavaScript.....	810
Scheduler with no timezone	810
Scheduler set to specific timezone	812
Display events on same time everywhere with no time difference	813
Set specific timezone for events	814
Add or remove timezone names to/from the timezone collection.....	816
Timezone methods	817
Views in EJ2 JavaScript Schedule control.....	819
Setting specific view on scheduler	820
View specific configuration	822
Extending view intervals	843
See Also	846

Calendar mode in EJ2 JavaScript Schedule control.....	846
Gregorian Calendar	846
Islamic Calendar	846
Resources in EJ2 JavaScript Schedule control.....	848
Resource fields.....	848
Resource data binding	849
Scheduler with multiple resources	851
Resource grouping	853
Customizing parent resource cells	863
Working with shared events	864
Simple resource header customization	866
Customizing resource header with multiple columns	869
Collapse/Expand child resources in timeline views	873
Displaying tooltip for resource headers.....	875
Choosing among resource colors for appointments.....	877
Setting different style to each resource appointments.....	879
Dynamically add and remove resources	881
Setting different working days and hours for resources	884
Hide non-working days when grouped by date.....	887
Scroll to specific resource	889
Compact view in mobile.....	890
Adaptive UI in desktop.....	892
Header rows in EJ2 JavaScript Schedule control.....	894
Display year and month rows in timeline views	896
Display week numbers in timeline views.....	897
Timeline view displaying dates of a complete year	899
Customizing the header rows using template	900
Row auto height in EJ2 JavaScript Schedule control.....	902
Calendar month view	902
Timeline views.....	903
Timeline views with multiple resources	905
Appointments occupying entire cell	907
Header bar in EJ2 JavaScript Schedule control	908
Show or Hide header bar	908
Customizing header bar using template	910

Customizing header bar using events	912
How to display the view options within the header bar popup	914
Date header customization	915
Customizing the date range text	919
Customizing header indent cells	920
Timescale in EJ2 JavaScript Schedule control	923
Setting different time slot duration	923
Customizing time cells using template	924
Hide the timescale	926
Highlighting current date and time	928
Working days in EJ2 JavaScript Schedule control	929
Set working days	929
Hiding weekend days	931
Show week numbers	932
Set working hours	935
Scheduler displaying custom hours	937
Setting start day of the week	938
Scroll to specific time and date	939
See Also	942
Cell customization in EJ2 JavaScript Schedule control	942
Setting cell dimensions in all views	942
Check for cell availability	944
Customizing cells in all the views	945
Customizing cell header in month view	949
Customizing the minimum and maximum date values	950
How to disable multiple cell and row selection in Schedule	952
State persistence in EJ2 JavaScript Schedule control	952
Exporting in EJ2 JavaScript Schedule control	953
Excel Exporting	954
Exporting calendar events as ICS file	968
Import events from other calendars	970
How to print the Scheduler element	972
Context menu in EJ2 JavaScript Schedule control	976
Dimensions in EJ2 JavaScript Schedule control	980
Auto Height and Width	980

Height and Width in pixel	981
Height and Width in percentage.....	982
See Also	984
Recurrence editor in EJ2 JavaScript Schedule control	984
Customizing the repeat type option in editor	984
Customizing the End Type Option in Editor	986
Accessing the recurrence rule string.....	987
Set specific value on recurrence editor	989
Recurrence date generation	990
Recurrence date generation in server-side.....	993
Restrict date generation with specific count	993
Localization in EJ2 JavaScript Schedule control	995
Globalization	995
Localizing the static Scheduler text.....	998
Setting date format.....	1003
Setting the time format	1005
Displaying Scheduler in RTL mode	1006
See Also	1007
Accessibility in EJ2 JavaScript Schedule control.....	1007
ARIA attributes.....	1008
Keyboard interaction	1009
Ensuring accessibility	1010
See also	1010
Scheduler styling in EJ2 JavaScript Schedule control.....	1010
Frequently asked questions in EJ2 JavaScript Schedule control.....	1012
Grouping with empty resources	1012
Not providing e-field in editor template.....	1012
Missing CSS reference	1012
QuickInfoTemplate at bottom	1013
Not processing culture files while using localization	1014
Getting instance of the Scheduler component.....	1015
Ej1 api migration in EJ2 JavaScript Schedule control	1015
Scheduler	1015
Properties.....	1015
Methods.....	1025

Events.....	1028
How To	1031
Add edit and remove events in EJ2 JavaScript Schedule control.....	1031
Set default value for event fields in EJ2 JavaScript Schedule control.....	1035
Open event editor manually in EJ2 JavaScript Schedule control.....	1039
Prevent date navigation in EJ2 JavaScript Schedule control.....	1042
Half yearly view in EJ2 JavaScript Schedule control.....	1045
Set different work hours in EJ2 JavaScript Schedule control.....	1048
Zoom in and zoom out the schedule in EJ2 JavaScript Schedule control	1049
Show entire time in responsive mode in EJ2 JavaScript Schedule control	1052
Quick info template in EJ2 JavaScript Schedule control	1054
Enable scroll option on all day section in EJ2 JavaScript Schedule control	1059
Manual refresh in EJ2 JavaScript Schedule control	1060
Sidebar	1062
Getting started in EJ2 JavaScript Sidebar control	1062
Component Initialization.....	1062
Enable backdrop	1066
Position	1068
Animate.....	1070
Close on document click	1071
Enable gestures.....	1073
See Also	1074
Custom context in EJ2 JavaScript Sidebar control	1075
Variations in EJ2 JavaScript Sidebar control	1076
Expanding types of Sidebar.....	1077
See Also	1080
Auto close in EJ2 JavaScript Sidebar control.....	1080
Docking sidebar in EJ2 JavaScript Sidebar control	1082
See Also	1084
Style in EJ2 JavaScript Sidebar control.....	1084
Customizing the sidebar.....	1084
Customizing the sidebar based on the positions	1084
Customizing the sidebar based on the active state	1085
Customizing the sidebar with dock state.....	1086
Customizing the different types of sidebar.....	1086

Customizing the backdrop of the sidebar	1087
Customizing the sidebar in the RTL direction	1087
Prevent the animation transition for the Sidebar component	1087
Accessibility in EJ2 JavaScript Sidebar component	1087
WAI-ARIA attributes	1088
Keyboard interaction	1088
Ensuring accessibility	1089
See also	1089
How To	1089
Sidebar with list view in EJ2 JavaScript Sidebar control	1089
Open and close the sidebar in EJ2 JavaScript Sidebar control	1091
Top and bottom sidebar in EJ2 JavaScript Sidebar control	1093
Multiple sidebar in EJ2 JavaScript Sidebar control	1095
Multiple sidebar in same position in EJ2 JavaScript Sidebar control	1097
Sidebar with treeview in EJ2 JavaScript Sidebar control	1099
Fixed sidebar in EJ2 JavaScript Sidebar control	1103
Sidebar with variation animation in EJ2 JavaScript Sidebar control	1109
Signature	1111
Customization in EJ2 JavaScript Signature control	1111
Stroke Width	1112
Stroke Color	1113
Background Color	1115
Background Image	1117
See Also	1118
Open save in EJ2 JavaScript Signature control	1119
Open Signature	1119
Save Signature	1120
Save with Background	1126
Draw in EJ2 JavaScript Signature control	1129
Draw	1129
User interaction in EJ2 JavaScript Signature control	1132
Undo and Redo	1132
Clear	1133
Disabled	1133
ReadOnly	1133

Accessibility in EJ2 JavaScript Signature control.....	1136
Keyboard interaction	1137
Ensuring accessibility	1137
See also	1137
How To	1137
Toolbar integration in EJ2 JavaScript Signature control	1137
Skeleton	1146
Shapes in EJ2 JavaScript Skeleton control	1146
Circle skeleton shape	1146
Square skeleton shape	1146
Rectangle skeleton shape	1146
Text skeleton shape	1147
Shimmer effect in EJ2 JavaScript Skeleton control	1148
Styles in EJ2 JavaScript Skeleton control	1150
cssClass.....	1150
Visible.....	1151
Accessibility in EJ2 JavaScript Skeleton control	1152
ARIA attributes.....	1152
Smithchart.....	1152
Getting started in EJ2 JavaScript Smithchart control.....	1152
Dependencies.....	1152
Installation and Configuration	1152
Add Smith Chart to the Project.....	1154
Module Injection.....	1155
Add Series to Smithchart	1156
Add title to SmithChart	1158
Enable Marker to Smithchart.....	1159
Enable DataLabel to Smithchart Marker.....	1161
Enable Legend for Smithchart.....	1163
Enable Tooltip for Smithchart Series	1165
Working with data in EJ2 JavaScript Smithchart control	1167
Data Binding.....	1167
Smith chart dimensions in EJ2 JavaScript Smithchart control	1169
Size for Container.....	1169
Size for Smithchart.....	1171

Title subtitle in EJ2 JavaScript Smithchart control	1173
Enable title	1173
Title trim.....	1175
Smith chart axis in EJ2 JavaScript Smithchart control	1177
Labels Customization	1177
Gridlines	1179
Axisline	1181
Smith chart legend in EJ2 JavaScript Smithchart control.....	1182
Position and Alignment.....	1182
Customization	1188
Toggle Visibility	1193
Smith chart tooltip in EJ2 JavaScript Smithchart control.....	1195
Smith chart marker in EJ2 JavaScript Smithchart control	1197
Marker.....	1197
Datalabels	1200
Smith chart series in EJ2 JavaScript Smithchart control	1203
points or datasource	1203
Series customization	1205
Smith chart print in EJ2 JavaScript Smithchart control.....	1206
Print.....	1206
Export.....	1208
Accessibility in EJ2 JavaScript Smithchart control	1209
WAI-ARIA attributes.....	1210
Keyboard interaction	1210
Ensuring accessibility	1210
See also	1210
Sparkline	1211
Sparkline dimensions in EJ2 JavaScript Sparkline control.....	1211
Size for container	1211
Size for sparkline	1212
Sparkline types in EJ2 JavaScript Sparkline control	1214
Axis customization in EJ2 JavaScript Sparkline control.....	1220
Change value type of the sparkline.....	1220
Change min and max values of axis	1223
Change value of axis.....	1225

Axis line customization	1226
Special points customization in EJ2 JavaScript Sparkline control	1227
Range band in EJ2 JavaScript Sparkline control	1229
Range band customization.....	1230
Multiple range band customization	1231
Marker in EJ2 JavaScript Sparkline control	1232
Adding marker to the sparkline	1232
Adding marker to special point.....	1233
Customizing markers.....	1234
Data labels in EJ2 JavaScript Sparkline control	1235
Enable data label.....	1235
Customize data label.....	1236
Format data label text.....	1237
User interaction in EJ2 JavaScript Sparkline control.....	1238
Tooltip	1239
Track line	1243
Appearance in EJ2 JavaScript Sparkline control.....	1244
Sparkline border.....	1244
Sparkline padding.....	1245
Sparkline area customization.....	1246
Sparkline theme	1247
Localization in EJ2 JavaScript Sparkline control	1248
Tooltip format	1249
RTL.....	1250
Accessibility in EJ2 JavaScript Sparkline control	1251
WAI-ARIA attributes	1252
Keyboard interaction	1252
Ensuring accessibility	1252
See also	1252
Ej1 api migration in EJ2 JavaScript Sparkline control.....	1252
Sparkline Types	1252
Databinding.....	1252
Markers	1253
Data labels.....	1253
Range band	1255

Special points customization	1255
Axis customization	1256
Appearance customization	1257
Tooltip	1257
Rendering.....	1259
Localization	1259
Methods.....	1259
Events.....	1259
SpeedDial	1261
Items in EJ2 JavaScript Speed dial control	1261
Icons in SpeedDial items	1261
Animation.....	1266
Template	1268
Positions in EJ2 JavaScript Speed dial control.....	1268
Opens items on hover	1269
Programmatically show/hide SpeedDial items	1270
Programmatically refresh the position	1272
Display modes in EJ2 JavaScript Speed dial control.....	1273
Linear display mode	1273
Radial display mode (Radial Menu)	1275
Radial menu in EJ2 JavaScript Speed dial control	1275
Radial menu direction	1275
Radial menu start and end angle	1276
Offset.....	1277
Template in EJ2 JavaScript Speed dial control.....	1279
Item template	1279
Popup template	1281
Styles in EJ2 JavaScript Speed dial control.....	1283
SpeedDial button	1283
Disabled.....	1287
cssClass.....	1287
Visible.....	1289
Tooltip	1289
Opens on hover.....	1290
Modal in EJ2 JavaScript Speed dial control.....	1292

Event in EJ2 JavaScript Speed dial control	1293
clicked	1293
created	1294
beforeOpen	1294
onOpen	1295
beforeClose	1296
onClose.....	1296
beforeItemRender	1297
Accessibility in EJ2 JavaScript Speed dial control.....	1299
Keyboard interaction	1299
ARIA attributes.....	1300
Spinner	1300
Template in EJ2 JavaScript Spinner control	1300
Types in EJ2 JavaScript Spinner control	1302
Style in EJ2 JavaScript Spinner control.....	1304
Customizing the spinner	1304
SplitButton	1305
Icons and separator in EJ2 JavaScript Split button control	1305
SplitButton icons	1305
Separator.....	1310
See Also	1313
Popup items in EJ2 JavaScript Split button control.....	1313
Icons	1313
Template	1316
See Also	1323
Accessibility in EJ2 JavaScript Split button control	1323
WAI-ARIA attributes.....	1324
Keyboard interaction	1324
Ensuring accessibility	1324
See also	1325
How To	1325
Create right to left splitbutton in EJ2 JavaScript Split button control	1325
Group items in popup in EJ2 JavaScript Split button control.....	1327
Open a dialog on popup item click in EJ2 JavaScript Split button control.....	1329
Set the disabled state in EJ2 JavaScript Split button control	1331

Underline a character in a text in EJ2 JavaScript Split button control.....	1333
Ej1 api migration in EJ2 JavaScript Split button control.....	1335
Properties.....	1335
Methods.....	1336
Events.....	1337
Splitter.....	1338
Pane sizing in EJ2 JavaScript Splitter control	1338
Auto size panes	1338
Fixed pane	1339
Pane content in EJ2 JavaScript Splitter control.....	1341
HTML Markup	1342
JavaScript UI controls.....	1344
Plain content	1344
Pane content using selector.....	1345
Split panes in EJ2 JavaScript Splitter control	1347
Horizontal layout.....	1347
Vertical layout	1348
Separator.....	1350
Nested Splitter	1351
Add or remove pane	1353
See Also	1356
Expand and collapse in EJ2 JavaScript Splitter control	1356
Collapsible panes	1356
Programmatically control the expand and collapse action	1358
Specify initial state to panes	1359
See Also	1361
Resizing in EJ2 JavaScript Splitter control	1361
Min and Max validation	1361
Prevent resizing.....	1362
Refresh content on resizing	1364
Customize the resize grip and cursor.....	1364
See Also	1365
Different layouts in EJ2 JavaScript Splitter control.....	1365
Code editor style layout.....	1365
Outlook style layout.....	1369

target {	1373
groupedList.e-listview .e-list-group-item {	1373
splitter1 .settings.e-list-wrapper.e-list-multi-line.e-list-avatar {	1373
buttonSection {	1373
See Also	1374
Style in EJ2 JavaScript Splitter control	1374
Customizing the split bar	1374
Customizing the split bar resize handle	1374
Customizing the split bar arrows	1375
To hide the resize handle in Splitter	1376
Accessibility in EJ2 JavaScript Splitter control	1377
Keyboard interaction	1378
Ensuring accessibility	1378
See also	1378
Ej1 api migration in EJ2 JavaScript Splitter control.....	1378
Common.....	1378
Accessibility and Localization.....	1379
Control State	1380
State Maintenance.....	1380
Pane Properties.....	1380
Animation.....	1381
Spreadsheet	1382
Overview of the EJ2 JavaScript Spreadsheet control.....	1382
Key features	1382
Getting started in EJ2 JavaScript Spreadsheet control	1383
Dependencies.....	1383
Setup for local development.....	1383
Adding Syncfusion resources	1384
Add Spreadsheet control	1388
Run the application	1390
See Also	1392
Data binding in EJ2 JavaScript Spreadsheet control	1392
Local data	1392
Remote data.....	1393
Cell data binding	1399

Dynamic data binding and Datasource change event	1401
See Also	1404
Open save in EJ2 JavaScript Spreadsheet control.....	1404
Open.....	1404
Supported file formats	1411
Save	1411
Server Configuration	1423
Server Dependencies	1424
See Also	1424
Worksheet in EJ2 JavaScript Spreadsheet control.....	1424
Add sheet	1424
Delete sheet	1426
Rename sheet	1426
Headers	1427
Gridlines	1427
Sheet visibility	1429
See Also	1431
Cell range in EJ2 JavaScript Spreadsheet control.....	1431
Wrap text	1431
Merge cells.....	1433
Data Validation.....	1436
Auto Fill	1440
Clear	1443
See Also	1446
Editing in EJ2 JavaScript Spreadsheet control	1446
Edit cell.....	1446
Save cell.....	1446
Cancel editing.....	1447
Limitations.....	1449
See Also	1449
Formulas in EJ2 JavaScript Spreadsheet control.....	1449
Usage.....	1449
Create User Defined Functions / Custom Functions.....	1449
Formula bar	1456
Named Ranges	1457

Supported Formulas.....	1459
Formula Error Dialog.....	1462
See Also	1463
Formatting in EJ2 JavaScript Spreadsheet control.....	1463
Number Formatting	1463
Text and cell formatting.....	1468
Conditional Formatting	1472
See Also	1475
Freeze pane in EJ2 JavaScript Spreadsheet control	1475
Apply freezepanes on UI	1475
FrozenRows	1476
FrozenColumns	1476
Limitations.....	1478
See Also	1478
Context menu in EJ2 JavaScript Spreadsheet control.....	1478
Context Menu Items in Row Cell.....	1478
Context Menu Items in Row Header / Column Header	1478
Context Menu Items in Pager	1479
Context Menu Customization	1479
See Also	1484
Template in EJ2 JavaScript Spreadsheet control	1484
See Also	1488
Illustrations in EJ2 JavaScript Spreadsheet control.....	1488
Image.....	1488
Chart.....	1527
See Also	1534
Rows and columns in EJ2 JavaScript Spreadsheet control	1534
Insert	1535
Delete.....	1539
Limitations of insert and delete	1541
Hide and show	1541
Size	1543
Changing text in column headers	1546
See Also	1548
Filter in EJ2 JavaScript Spreadsheet control	1548

Apply filter on UI	1548
Filter by criteria	1548
Filter by cell value	1550
Clear filter.....	1550
Clear filter on a field.....	1550
Reapply filter	1551
Known error validations.....	1551
Limitations.....	1551
See Also	1551
Sort in EJ2 JavaScript Spreadsheet control	1551
Sort by cell value	1551
Data contains header	1553
Case sensitive sort.....	1554
Sort multiple columns	1554
Custom sort comparer	1557
Known error validations.....	1557
Limitations.....	1557
See Also	1557
Link in EJ2 JavaScript Spreadsheet control	1557
Insert Link.....	1557
Edit Hyperlink.....	1558
Remove Hyperlink.....	1558
How to change target attribute	1558
Limitations.....	1561
See Also	1561
Clipboard in EJ2 JavaScript Spreadsheet control	1561
Cut	1561
Copy	1562
Paste.....	1562
Prevent the paste functionality	1564
Limitations.....	1567
Scrolling in EJ2 JavaScript Spreadsheet control	1567
Finite Scrolling.....	1567
Virtual Scrolling.....	1567
Finite scrolling with defined rows and columns	1568

Selection in EJ2 JavaScript Spreadsheet control.....	1570
Cell selection	1570
Row selection	1570
Column selection	1572
How to remove selection in the spreadsheet.....	1574
Limitations.....	1576
Protect sheet in EJ2 JavaScript Spreadsheet control	1576
Protect Sheet	1576
Unprotect Sheet.....	1578
Unlock the particular cells in the protected sheet.....	1579
Protect Workbook.....	1581
Unprotect Workbook	1584
See Also	1584
Searching in EJ2 JavaScript Spreadsheet control	1584
Find.....	1584
Replace.....	1585
Go to.....	1585
Limitations.....	1587
Keyboard shortcuts in EJ2 JavaScript Spreadsheet control	1587
See Also	1590
Ribbon in EJ2 JavaScript Spreadsheet control	1590
Ribbon Customization	1590
See Also	1594
Global local in EJ2 JavaScript Spreadsheet control.....	1594
Localization	1594
Internationalization.....	1607
Right to left (RTL)	1610
See Also	1612
Undo redo in EJ2 JavaScript Spreadsheet control	1612
Undo.....	1612
Redo	1612
Update custom actions in UndoRedo collection.....	1612
See Also	1615
Accessibility in EJ2 JavaScript Spreadsheet control	1615
WAI-ARIA attributes.....	1616

Keyboard interaction	1616
Ensuring accessibility	1618
See also	1618
Styles in EJ2 JavaScript Spreadsheet control	1618
Customizing the Spreadsheet	1618
Header.....	1618
Sheet	1619
Ribbon Items.....	1620
Footer.....	1622
Use Cases	1622
Collaborative editing in EJ2 JavaScript Spreadsheet control	1622
How To	1628
Sort a range by custom list in EJ2 JavaScript Spreadsheet control.....	1628
Create a object structure in EJ2 JavaScript Spreadsheet control	1630
Print in EJ2 JavaScript Spreadsheet control	1635
Mobile responsiveness in EJ2 JavaScript Spreadsheet control.....	1650
Stepper.....	1651
Steps in EJ2 JavaScript Stepper control	1651
Adding steps.....	1651
Optional steps	1655
Disabling steps	1658
Setting active step.....	1660
Step status.....	1663
Step styling.....	1666
Step validation	1669
Step types in EJ2 JavaScript Stepper control	1669
Default type	1669
Label type.....	1672
Indicator type.....	1678
Orientations in EJ2 JavaScript Stepper control.....	1679
Horizontal.....	1679
Vertical	1682
Linear flow in EJ2 JavaScript Stepper control	1684
Steps validation in EJ2 JavaScript Stepper control.....	1687
Template in EJ2 JavaScript Stepper control.....	1690

Tooltip in EJ2 JavaScript Stepper control.....	1693
Tooltip template	1696
Animation in EJ2 JavaScript Stepper control	1699
Globalization in EJ2 JavaScript Stepper control	1700
Localization	1700
RTL.....	1703
Accessibility in EJ2 JavaScript Stepper control.....	1706
Keyboard interaction	1706
ARIA attribute	1706
Events in EJ2 JavaScript Stepper control	1706
created	1706
stepChanged	1707
stepChanging.....	1708
stepClick	1709
beforeStepRender.....	1710
Stock Chart.....	1711
Working with data in EJ2 JavaScript Stock chart control.....	1711
Local Data.....	1711
See Also	1712
Chart dimensions in EJ2 JavaScript Stock chart control.....	1713
Size for Container.....	1713
Size for Stock Chart	1714
Axis types in EJ2 JavaScript Stock chart control.....	1716
DateTime axis.....	1716
DateTimeCategory axis	1718
Logarithmic axis	1720
See also	1721
Axis customization in EJ2 JavaScript Stock chart control.....	1722
Axis Crossing	1722
Title	1723
Tick Lines Customization	1724
Grid Lines Customization	1726
Multiple Axis	1727
Inversed Axis	1729
Opposed Position	1730

Series types in EJ2 JavaScript Stock chart control.....	1732
Line.....	1732
Spline.....	1732
Hilo	1732
HiloOpenClose.....	1732
HollowCandle	1732
Candle	1732
Trend lines in EJ2 JavaScript Stock chart control	1733
Linear.....	1733
Exponential	1733
Logarithmic	1734
Polynomial	1734
Power.....	1734
Moving Average	1734
Technical indicators in EJ2 JavaScript Stock chart control.....	1737
Accumulation Distribution	1737
Average True Range (ATR)	1737
Exponential Moving Average (EMA)	1738
Momentum	1738
Moving Average Convergence Divergence (MACD)	1738
Relative Strength Index (RSI).....	1738
Simple Moving Average (SMA)	1738
Stochastic	1738
Triangular Moving Average (TMA).....	1738
Bollinger Band.....	1738
Tool tip in EJ2 JavaScript Stock chart control.....	1740
Default tooltip.....	1740
Format the tooltip.....	1742
Position the tooltip	1743
Customize the appearance of the tooltip	1745
Cross hair in EJ2 JavaScript Stock chart control	1746
Tooltip for axis	1747
Customization	1749
Legend in EJ2 JavaScript Stock chart control	1750
Position and Alignment.....	1750

Customization	1755
Collapsing Legend Item	1759
Legend Title	1761
Period selector in EJ2 JavaScript Stock chart control	1763
Periods	1763
Visibility of period selector	1765
Range selector in EJ2 JavaScript Stock chart control	1766
Export print in EJ2 JavaScript Stock chart control.....	1768
Disable Export and print	1769
Appearance in EJ2 JavaScript Stock chart control	1770
Stock Chart Title	1770
Stock Chart Theme	1773
See Also	1775
Stock events in EJ2 JavaScript Stock chart control	1775
See Also	1781
Accessibility in EJ2 JavaScript Stock chart control	1781
WAI-ARIA attributes.....	1782
Keyboard interaction	1782
Ensuring accessibility	1783
See also	1783
Switch.....	1783
Accessibility in EJ2 JavaScript Switch control.....	1783
WAI-ARIA attributes.....	1784
Keyboard interaction	1784
Ensuring accessibility	1784
See also	1784
How To	1784
Change size in EJ2 JavaScript Switch control	1784
Customize the appearance of a switch in EJ2 JavaScript Switch control.....	1786
Enable ripple for switch label in EJ2 JavaScript Switch control	1792
Enable rtl in EJ2 JavaScript Switch control.....	1794
Set disabled state in EJ2 JavaScript Switch control.....	1795
Submit name and value in form in EJ2 JavaScript Switch control	1797
Change switch state using toggle method in EJ2 JavaScript Switch control.....	1799
Tabs.....	1800

Getting started in EJ2 JavaScript Tab control	1800
Component Initialization.....	1800
Initialize the Tab using JSON items collection.....	1801
Initialize the Tab using HTML elements	1805
Adaptive in EJ2 JavaScript Tab control.....	1807
Scrollable.....	1807
Popup	1810
See Also	1813
Header in EJ2 JavaScript Tab control	1813
Styles	1813
Icon positions	1816
See Also	1818
Localization in EJ2 JavaScript Tab control	1818
Loading translations	1819
Orientation in EJ2 JavaScript Tab control	1821
Drag and drop in EJ2 JavaScript Tab control	1825
Drag and drop item between tabs	1827
Drag and drop items to external source	1830
Drag and drop items from external source.....	1833
Accessibility in EJ2 JavaScript Tab control	1836
ARIA attributes	1837
Keyboard interaction	1837
Ensuring accessibility	1838
See also	1838
Style in EJ2 JavaScript Tab control	1838
Customizing Tab	1838
Customizing the Tab items.....	1839
Customizing Tab's header	1839
Customizing Tab's header icon	1839
Customizing Tab's content	1839
Customizing the hover state of Tab control.....	1840
Customizing selected item of Tab control	1840
How To	1840
Load content through post in EJ2 JavaScript Tab control	1840
Prevent content swipe selection in EJ2 JavaScript Tab control	1842

Customize selected tab styles in EJ2 JavaScript Tab control	1844
Customize tab scroll step in EJ2 JavaScript Tab control.....	1845
Create wizard using tab in EJ2 JavaScript Tab control	1848
Load tab with data source in EJ2 JavaScript Tab control	1858
Add nested tabs in EJ2 JavaScript Tab control.....	1859
Add font awesome in EJ2 JavaScript Tab control	1860
Set state persistence of the tab component in EJ2 JavaScript Tab control	1862
Set custom animation in EJ2 JavaScript Tab control.....	1864
Load tab items dynamically in EJ2 JavaScript Tab control	1867
Create collapsible tabs in EJ2 JavaScript Tab control.....	1869
Customize tab content height in EJ2 JavaScript Tab control	1872
Reorder active tab in EJ2 JavaScript Tab control	1874
Display tool tip on tab header in EJ2 JavaScript Tab control	1878
Tab selection in EJ2 JavaScript Tab control.....	1879
Tab key navigation in EJ2 JavaScript Tab control.....	1882
TextBox	1885
Groups in EJ2 JavaScript Textbox control	1885
With icon and floating label	1886
With clear button and floating label	1890
Floating label without required attribute	1892
Multi-line input with floating label	1894
See Also	1895
Sizing in EJ2 JavaScript Textbox control.....	1896
Validation in EJ2 JavaScript Textbox control.....	1899
Adding mandatory asterisk to placeholder and float label.....	1902
Multiline in EJ2 JavaScript Textbox control	1904
Create multiline textbox	1904
Implementing floating label	1906
Auto resizing	1907
Disable resizing	1909
Limit the text length.....	1911
Count characters.....	1912
Style appearance in EJ2 JavaScript Textbox control	1914
Filled and Outline mode.....	1916
How To	1917

Set the rounded corner in EJ2 JavaScript Textbox control	1917
Set the disabled state in EJ2 JavaScript Textbox control	1920
Set the read only textbox in EJ2 JavaScript Textbox control	1922
Add textbox programmatically in EJ2 JavaScript Textbox control	1924
Add floating label to textbox programmatically in EJ2 JavaScript Textbox control.....	1927
Change the floating label color of the textbox in EJ2 JavaScript Textbox control	1931
Change the color of the textbox based on its value in EJ2 JavaScript Textbox control	1934
Add floating label to read only textbox in EJ2 JavaScript Textbox control	1937
Customize the textbox background color and text color in EJ2 JavaScript Textbox control	1938
Migration from css textbox to javascript textbox in EJ2 JavaScript Textbox control	1940
Normal textbox	1940
Floating label textbox.....	1941
TimePicker.....	1942
Time range in EJ2 JavaScript Timepicker control	1942
Time Range customization using two TimePicker components	1943
Time masking in EJ2 JavaScript Timepicker control	1945
Configure Mask Placeholder	1948
Globalization in EJ2 JavaScript Timepicker control.....	1949
Right-To-Left	1954
Strict mode in EJ2 JavaScript Timepicker control	1956
Accessibility in EJ2 JavaScript Timepicker control	1958
WAI-ARIA attributes.....	1959
Keyboard Interaction	1960
Ensuring accessibility	1962
See also	1962
Style appearance in EJ2 JavaScript Timepicker control	1962
Customizing the appearance of TimePicker wrapper element	1962
Customizing the TimePicker icon element.....	1962
Customizing the TimePicker popup	1962
Customizing the TimePicker popup content.....	1963
Full screen mode support in mobiles and tablets.....	1963
How To	1965
Css customization in EJ2 JavaScript Timepicker control	1965
Client side validation using form validator in EJ2 JavaScript Timepicker control.....	1966
Ej1 api migration in EJ2 JavaScript Timepicker control.....	1968

Time Selection.....	1968
Time Format.....	1968
Time Range.....	1968
Disabled Time Ranges	1969
Customization	1969
Accessibility.....	1971
Persistence.....	1971
Validation	1972
Common.....	1972
Globalization	1974
Strict Mode	1974
Open and Close	1975
Toast.....	1975
Config in EJ2 JavaScript Toast control.....	1975
Title and content template	1976
Specifying custom target	1977
Close button.....	1977
Progress bar	1977
Newest on top.....	1977
Width and height	1978
See Also	1981
Position in EJ2 JavaScript Toast control.....	1981
Predefined.....	1981
Custom	1981
Action buttons in EJ2 JavaScript Toast control	1986
See Also	1987
Timeout in EJ2 JavaScript Toast control.....	1987
Static toast	1989
See Also	1990
Template in EJ2 JavaScript Toast control.....	1990
See Also	1992
Animation in EJ2 JavaScript Toast control	1992
Toast services in EJ2 JavaScript Toast control	1995
Show Toast with predefined types	1995
Show Toast with ToastModel	1997

Style in EJ2 JavaScript Toast control	1998
Customizing the toast title	1998
Customizing the toast content.....	1998
Customizing the toast icon.....	1998
Customizing the toast background	1999
Accessibility in EJ2 JavaScript Toast control	1999
WAI-ARIA attributes.....	2000
Ensuring accessibility	2001
See also	2001
How To	2001
Prevent duplicate toast display in EJ2 JavaScript Toast control	2001
Restrict the maximum toast to show in EJ2 JavaScript Toast control	2003
Customize progress bar theme and sizing in EJ2 JavaScript Toast control.....	2005
Play an audio before open the toast in EJ2 JavaScript Toast control	2007
Show different types of toast in EJ2 JavaScript Toast control	2008
Show multiple toasts in various positions in EJ2 JavaScript Toast control	2009
Close the toast with click tap in EJ2 JavaScript Toast control.....	2011
Add dynamic template in EJ2 JavaScript Toast control	2012
Prevent toast close with mobile swipe in EJ2 JavaScript Toast control.....	2013
Toolbar	2014
Getting started in EJ2 JavaScript Toolbar control.....	2014
Component Initialization.....	2014
Initialize the Toolbar with commands	2016
See Also	2019
Item configuration in EJ2 JavaScript Toolbar control	2019
Button	2019
Separator.....	2019
Input.....	2020
See Also	2025
Responsive mode in EJ2 JavaScript Toolbar control	2025
Scrollable.....	2025
Popup	2028
Template configuration in EJ2 JavaScript Toolbar control.....	2030
Popup customization	2031
Integrate menu component.....	2032

Accessibility in EJ2 JavaScript Toolbar control.....	2035
ARIA attributes.....	2036
Keyboard interaction	2036
Ensuring accessibility	2037
See also	2037
Style in EJ2 JavaScript Toolbar control.....	2037
Customizing Toolbar	2037
Customizing the Toolbar items	2037
Customizing Toolbar's item icon.....	2038
Customizing the hover state of Toolbar control.....	2038
Customizing selected item of Toolbar control.....	2038
How To	2038
Set command customization in EJ2 JavaScript Toolbar control.....	2038
Set tool tip to the commands in EJ2 JavaScript Toolbar control	2040
Set item wise custom template in EJ2 JavaScript Toolbar control	2041
Add font awesome in EJ2 JavaScript Toolbar control	2042
Add toggle button in EJ2 JavaScript Toolbar control	2044
Enable or disable toolbar item in EJ2 JavaScript Toolbar control.....	2046
How to customize toolbar scroll step in EJ2 JavaScript Toolbar control	2049
Add link to toolbar item in EJ2 JavaScript Toolbar control	2051
Ej1 api migration in EJ2 JavaScript Toolbar control	2052
Accessibility.....	2052
DataSource.....	2053
Items	2053
Common.....	2056
Tooltip	2058
Getting started in EJ2 JavaScript Tooltip control.....	2058
control Initialization	2058
See Also	2061
Setting dimension in EJ2 JavaScript Tooltip control	2061
Height and width.....	2061
Content in EJ2 JavaScript Tooltip control	2063
Template content.....	2063
Dynamic content via AJAX.....	2065
Position in EJ2 JavaScript Tooltip control	2066

Tip pointer positioning.....	2068
Dynamic positioning.....	2069
Mouse trailing.....	2070
Setting offset values.....	2071
Open mode in EJ2 JavaScript Tooltip control	2073
Custom open mode.....	2075
Sticky mode.....	2076
Open/Close Tooltip with delay	2077
Animation in EJ2 JavaScript Tooltip control.....	2078
Animation effects.....	2079
Animating via open/close methods	2080
Apply transition.....	2081
Customization in EJ2 JavaScript Tooltip control	2082
Tip pointer customization	2083
Tooltip customization	2083
Style in EJ2 JavaScript Tooltip control.....	2084
Customizing the tooltip.....	2084
Customizing the tooltip popup	2085
Customizing the tooltip content	2085
Customizing the tooltip arrow tip.....	2085
Customizing the tooltip inner tip	2086
Customizing the tooltip outer tip.....	2086
Accessibility in EJ2 JavaScript Tooltip component.....	2087
WAI-ARIA attributes.....	2088
Keyboard interaction	2089
Ensuring accessibility	2089
See also	2089
Ej1 api migration in EJ2 JavaScript Tooltip control	2089
How To	2091
Show tooltip on disabled elements and disable tooltip in EJ2 JavaScript Tooltip control	2091
Dynamic tooltip content with html in EJ2 JavaScript Tooltip control.....	2093
Custom tooltip with dynamic html in EJ2 JavaScript Tooltip control	2094
Tooltip open or display modes in EJ2 JavaScript Tooltip control.....	2096
Fancy tooltip customization in EJ2 JavaScript Tooltip control.....	2098
Display tooltip on svg and canvas elements in EJ2 JavaScript Tooltip control	2102

Dynamic tooltip content in EJ2 JavaScript Tooltip control	2105
Create and show tooltip on multiple targets in EJ2 JavaScript Tooltip control	2107
TreeGrid	2110
Dependencies.....	2110
Setup for local environment	2111
Adding Syncfusion resources	2111
Adding TreeGrid control	2114
Defining Row Data	2115
Defining Columns	2116
Module injection.....	2117
Enable paging.....	2117
Enable sorting	2119
Enable filtering.....	2121
Run the application	2122
Deploy the application.....	2124
Module in EJ2 JavaScript Treegrid control.....	2124
Data Binding.....	2125
Data binding in EJ2 JavaScript Treegrid control.....	2125
Local data in EJ2 JavaScript Treegrid control	2138
Remote data in EJ2 JavaScript Treegrid control	2142
Immutable mode in EJ2 JavaScript Treegrid control	2171
Limitations.....	2175
Columns	2175
Columns in EJ2 JavaScript Treegrid control	2175
Headers in EJ2 JavaScript Treegrid control	2192
Column template in EJ2 JavaScript Treegrid control	2196
Complex data binding in EJ2 JavaScript Treegrid control	2200
Auto fit columns in EJ2 JavaScript Treegrid control.....	2201
Column reorder in EJ2 JavaScript Treegrid control.....	2203
Column resizing in EJ2 JavaScript Treegrid control.....	2206
Column chooser in EJ2 JavaScript Treegrid control	2211
Column menu in EJ2 JavaScript Treegrid control.....	2215
Responsive columns in EJ2 JavaScript Treegrid control.....	2221
Row	2222
Row in EJ2 JavaScript Treegrid control	2222

Indent in EJ2 JavaScript Treegrid control.....	2226
Row height in EJ2 JavaScript Treegrid control	2229
Row template in EJ2 JavaScript Treegrid control.....	2233
Detail template in EJ2 JavaScript Treegrid control	2237
Row drag and drop in EJ2 JavaScript Treegrid control.....	2239
Cell	2245
Cell in EJ2 JavaScript Treegrid control	2245
Content in EJ2 JavaScript Treegrid control	2250
Auto wrap in EJ2 JavaScript Treegrid control	2252
Clip mode in EJ2 JavaScript Treegrid control	2254
Editing	2256
Edit in EJ2 JavaScript Treegrid control	2256
Edit types in EJ2 JavaScript Treegrid control	2263
Cell editing in EJ2 JavaScript Treegrid control	2267
Row editing in EJ2 JavaScript Treegrid control	2268
Dialog editing in EJ2 JavaScript Treegrid control	2270
Batch editing in EJ2 JavaScript Treegrid control	2271
Template editing in EJ2 JavaScript Treegrid control	2272
Command column editing in EJ2 JavaScript Treegrid control.....	2278
Validation in EJ2 JavaScript Treegrid control	2281
Persisting data in server in EJ2 JavaScript Treegrid control.....	2284
Sorting in EJ2 JavaScript Treegrid control.....	2290
Initial sort	2291
Sorting events	2293
Touch interaction	2297
Filtering	2297
Filtering in EJ2 JavaScript Treegrid control	2297
Filter bar in EJ2 JavaScript Treegrid control.....	2305
Filter menu in EJ2 JavaScript Treegrid control.....	2311
Excel like filter in EJ2 JavaScript Treegrid control	2316
Searching in EJ2 JavaScript Treegrid control.....	2319
Initial search	2321
Search operators	2323
Search by external button.....	2323
Search specific columns	2325

Selection.....	2326
Selection in EJ2 JavaScript Treegrid control.....	2326
Row selection in EJ2 JavaScript Treegrid control.....	2330
Cell selection in EJ2 JavaScript Treegrid control.....	2336
Check box selection in EJ2 JavaScript Treegrid control	2338
Paging in EJ2 JavaScript Treegrid control.....	2342
Page Size Mode	2343
Template	2345
Pager with Page Size Dropdown	2347
How to render Pager at the Top of the TreeGrid.....	2348
Scrolling in EJ2 JavaScript Treegrid control.....	2350
Set width and height.....	2350
Responsive with parent container	2352
Scroll to selected row.....	2354
Frozen in EJ2 JavaScript Treegrid control	2355
Frozen rows and columns	2355
Virtual scroll in EJ2 JavaScript Treegrid control	2361
Row virtualization	2361
Column virtualization.....	2363
Limitations for virtualization.....	2366
Infinite scroll in EJ2 JavaScript Treegrid control	2366
InitialBlocks	2368
Cache Mode	2370
Limitations for Infinite Scrolling.....	2371
Aggregates	2372
Aggregates in EJ2 JavaScript Treegrid control	2372
Footer aggregate in EJ2 JavaScript Treegrid control.....	2374
Custom aggregate in EJ2 JavaScript Treegrid control	2378
Print in EJ2 JavaScript Treegrid control.....	2380
Page setup.....	2381
Print using an external button	2381
Print the visible page.....	2383
Print large number of columns	2385
Show or Hide columns while Printing	2385
Limitations of Printing Large Data.....	2387

State Persistence.....	2388
State persistence in EJ2 JavaScript Treegrid control	2388
Get or set local storage value in EJ2 JavaScript Treegrid control	2388
Tool Bar	2388
Tool bar in EJ2 JavaScript Treegrid control	2388
Tool bar items in EJ2 JavaScript Treegrid control	2389
Custom tool bar in EJ2 JavaScript Treegrid control	2393
Pdf Export.....	2396
Pdf export in EJ2 JavaScript Treegrid control	2396
Pdf export options in EJ2 JavaScript Treegrid control	2404
Pdf cell style customization in EJ2 JavaScript Treegrid control	2417
Adding header and footer in EJ2 JavaScript Treegrid control.....	2421
Exporting Tree Grid in Server	2425
Excel Export.....	2429
Excel export in EJ2 JavaScript Treegrid control.....	2429
Excel export options in EJ2 JavaScript Treegrid control.....	2435
Excel cell style customization in EJ2 JavaScript Treegrid control.....	2441
Adding header and footer in EJ2 JavaScript Treegrid control.....	2445
Exporting Tree Grid in Server	2447
Global local in EJ2 JavaScript Treegrid control	2452
Localization	2452
Internationalization.....	2457
Right to left (RTL)	2460
See Also	2462
Accessibility in EJ2 JavaScript Treegrid control.....	2463
WAI-ARIA attributes.....	2464
Keyboard interaction	2464
Ensuring accessibility	2465
See also	2466
Clipboard in EJ2 JavaScript Treegrid control.....	2466
Copy to clipboard by external buttons	2467
Copy Hierarchy Modes.....	2468
AutoFill	2470
Paste.....	2472
Context menu in EJ2 JavaScript Treegrid control	2474

Custom context menu items	2476
Enable and disable context menu items dynamically	2478
Adaptive in EJ2 JavaScript Treegrid control	2481
Loading animation in EJ2 JavaScript Treegrid control	2486
Test Helpers	2488
Helpers in EJ2 JavaScript Treegrid control	2488
Cypress in EJ2 JavaScript Treegrid control	2488
Treegrid styling in EJ2 JavaScript Treegrid control	2492
How To	2493
Refresh the data source in EJ2 JavaScript Treegrid control	2493
Enable disable treegrid and its actions in EJ2 JavaScript Treegrid control	2495
Change header text dynamically in EJ2 JavaScript Treegrid control	2498
Customize column styles in EJ2 JavaScript Treegrid control	2500
Custom tool tip for columns in EJ2 JavaScript Treegrid control	2502
Change orientation of header text in EJ2 JavaScript Treegrid control	2504
Render other component in column in EJ2 JavaScript Treegrid control	2507
Customize the icon for column menu in EJ2 JavaScript Treegrid control	2509
Customize the edit dialog in EJ2 JavaScript Treegrid control	2511
Cascading drop down list with treegrid editing in EJ2 JavaScript Treegrid control	2513
Provide custom data source and enabling filtering to drop down list in EJ2 JavaScript Treegrid control	2516
Use wizard like dialog editing in EJ2 JavaScript Treegrid control	2519
Using tab inside the dialog editing in EJ2 JavaScript Treegrid control	2523
Restrict decimal points while treegrid editing in EJ2 JavaScript Treegrid control	2527
Exporting filtered data in EJ2 JavaScript Treegrid control	2529
Exporting selected data in EJ2 JavaScript Treegrid control	2531
Show spinner while exporting in EJ2 JavaScript Treegrid control	2533
Customize pager drop down in EJ2 JavaScript Treegrid control	2536
Add parameter for filtering in EJ2 JavaScript Treegrid control	2537
Select treegrid rows based on certain condition in EJ2 JavaScript Treegrid control	2539
Get row cell index in EJ2 JavaScript Treegrid control	2541
Display foreign key values in treegrid in EJ2 JavaScript Treegrid control	2543
Row cell customization in EJ2 JavaScript Treegrid control	2545
Treemap	2547
Data binding in EJ2 JavaScript Treemap control	2547

Populate data	2548
Layout in EJ2 JavaScript Treemap control.....	2551
Types of layout.....	2551
Leaf item in EJ2 JavaScript Treemap control	2556
Leaf label	2556
Item gap	2559
Levels in EJ2 JavaScript Treemap control.....	2560
Group path	2560
Group gap.....	2562
Header format and Alignment	2563
Header height and style	2565
Header template and position	2567
Color mapping in EJ2 JavaScript Treemap control.....	2568
Range color mapping	2568
Equal color mapping	2570
Desaturation color mapping	2571
Palette color mapping.....	2572
Desaturation with multiple colors	2573
Color for items excluded from color mapping	2574
Bind the colors to the items from data source	2576
Data label in EJ2 JavaScript Treemap control	2577
Format.....	2577
Template	2578
InterSectAction	2579
Legend in EJ2 JavaScript Treemap control.....	2580
Position and alignment	2580
Legend mode.....	2583
Legend size.....	2585
Legend for items excluded from color mapping.....	2588
Hide desired legend items	2589
Hide legend items based data source value	2591
Bind legend item text from data source	2592
Hide duplicate legend items	2593
Legend Responsiveness	2594
Drilldown in EJ2 JavaScript Treemap control.....	2595

Perform drill-down action	2595
On-demand data loading	2597
Breadcrumb support	2599
Tooltip in EJ2 JavaScript Treemap control	2600
Default tooltip	2600
Format tooltip	2601
Tooltip template	2602
Selection and highlight in EJ2 JavaScript Treemap control.....	2604
Selection.....	2604
Highlight	2605
Print and export in EJ2 JavaScript Treemap control	2607
Print.....	2607
Export.....	2608
Accessibility in EJ2 JavaScript TreeMap control.....	2612
WAI-ARIA attributes.....	2612
Screen reading in TreeMap.....	2613
Ensuring accessibility	2613
See also	2613
Internationalization in EJ2 JavaScript Treemap control.....	2613
Globalization	2613
Right-to-left rendering	2615
Legend with Rtl support.....	2615
Tooltip with Rtl support	2616
Treemap Item Rendering Direction	2617
Right-to-left rendering	2621
Legend with Rtl support.....	2622
Tooltip with Rtl support	2623
Treemap Item Rendering Direction	2624
Ej1 api migration in EJ2 JavaScript Treemap control	2628
Data Binding.....	2628
Appearance	2628
Leaf Items.....	2629
Legend.....	2630
Levels.....	2632
Selection.....	2634

Highlight.....	2634
Range ColorMapping.....	2635
Desaturation ColorMapping	2635
Tooltip	2636
Drilldown.....	2637
Methods.....	2637
Events.....	2638
How To	2639
Drilldown in EJ2 JavaScript Treemap control.....	2639
Drilldown with label in EJ2 JavaScript Treemap control	2643
TreeView	2645
Data binding in EJ2 JavaScript Treeview control	2645
Local data	2645
Remote data.....	2649
Check box in EJ2 JavaScript Treeview control.....	2651
Checked nodes	2653
See Also	2654
Node editing in EJ2 JavaScript Treeview control	2654
See Also	2657
Multiple selection in EJ2 JavaScript Treeview control	2657
Selected nodes	2659
Drag and drop in EJ2 JavaScript Treeview control	2661
Multiple-node drag and drop.....	2663
See Also	2665
Template in EJ2 JavaScript Treeview control.....	2665
See Also	2667
Accessibility in EJ2 JavaScript Treeview component	2667
WAI-ARIA attributes.....	2668
Keyboard interaction	2669
Ensuring accessibility	2669
See also	2669
How To	2669
Customize the expand and collapse icons in EJ2 JavaScript Treeview control.....	2669
Process the tree node operations using context menu in EJ2 JavaScript Treeview control.....	2673
Check uncheck the checkbox on clicking the tree node text in EJ2 JavaScript Treeview control ..	2676

Validate the text when renaming the tree node in EJ2 JavaScript Treeview control	2678
Customize the tree nodes based on levels in EJ2 JavaScript Treeview control	2680
Restrict the drag and drop for particular tree nodes in EJ2 JavaScript Treeview control	2683
Accordion tree in EJ2 JavaScript Treeview control	2687
Auto hide show expand collapse icon in EJ2 JavaScript Treeview control	2689
Filtering tree nodes in EJ2 JavaScript Treeview control.....	2692
Set tool tip for tree nodes in EJ2 JavaScript Treeview control	2695
Sorting treeview level wise in EJ2 JavaScript Treeview control.....	2698
Remove parent checkbox in EJ2 JavaScript Treeview control	2700
Get dynamic icon in EJ2 JavaScript Treeview control	2702
Hover multi line tree node in EJ2 JavaScript Treeview control	2704
Select one child in EJ2 JavaScript Treeview control.....	2707
Get all child nodes in EJ2 JavaScript Treeview control	2709
Disable checkbox of the tree node in EJ2 JavaScript Treeview control.....	2712
Ej1 api migration in EJ2 JavaScript Treeview control.....	2713
Add nodes	2713
Common.....	2714
CheckBox.....	2722
Drag and Drop.....	2724
Expand/Collapse nodes.....	2725
Node Editing.....	2727
Node Selection	2728
Template	2730
Uploader	2731
Async in EJ2 JavaScript Uploader control	2731
Multiple file upload.....	2731
Single file upload.....	2733
Save Action.....	2734
Remove Action	2737
Auto Upload	2740
Sequential Upload.....	2741
Preload Files.....	2742
Adding additional HTTP headers with upload action.....	2744
See Also	2745
Chunk upload in EJ2 JavaScript Uploader control.....	2745

Additional configurations.....	2747
Resumable upload	2749
Cancel upload.....	2750
Server-Side configurations.....	2752
File source in EJ2 JavaScript Uploader control.....	2756
Paste to upload	2756
Directory upload	2758
Drag and drop	2761
See Also	2764
Validation in EJ2 JavaScript Uploader control	2764
File type.....	2765
File size.....	2766
Maximum files count	2767
Duplicate files.....	2770
See Also	2771
Form support in EJ2 JavaScript Uploader control.....	2772
Template in EJ2 JavaScript Uploader control.....	2777
File list template.....	2777
Custom template	2785
See Also	2794
Localization in EJ2 JavaScript Uploader control.....	2794
Wai aria accessibility in EJ2 JavaScript Uploader control	2797
Keyboard interaction	2798
Ensuring accessibility	2800
See also	2800
Style appearance in EJ2 JavaScript Uploader control	2800
Customizing the appearance of File Upload wrapper element	2800
Customizing the File Upload browse button	2800
Customizing the File Upload content.....	2800
Customizing the uploaded file container in File Upload.....	2801
See Also	2801
How To	2801
Hide default drop area in EJ2 JavaScript Uploader control	2801
Preview images before uploading in EJ2 JavaScript Uploader control	2803
Add html attributes in EJ2 JavaScript Uploader control	2803

Achieve invisible upload in EJ2 JavaScript Uploader control	2805
Customize progressbar in EJ2 JavaScript Uploader control	2807
Sort the selected files in EJ2 JavaScript Uploader control	2809
Get the total size of selected files in EJ2 JavaScript Uploader control	2811
Customize button with html element in EJ2 JavaScript Uploader control	2812
Add additional data on upload in EJ2 JavaScript Uploader control	2816
Validate image on drop in EJ2 JavaScript Uploader control	2817
Determine whether the uploader has input file in EJ2 JavaScript Uploader control	2819
Achieve file upload programmatically in EJ2 JavaScript Uploader control	2823
Check file size before uploading in EJ2 JavaScript Uploader control	2825
Check the mime type of file before upload in EJ2 JavaScript Uploader control	2827
Trigger click event of input file from external button in EJ2 JavaScript Uploader control	2828
Open and edit the uploaded files in EJ2 JavaScript Uploader control	2830
Resize images before uploading it to the server in EJ2 JavaScript Uploader control	2832
upload {	2833
Convert image into binary format after uploading in EJ2 JavaScript Uploader control	2841
Ej1 api migration in EJ2 JavaScript Uploader control	2842
Accessibility	2842
File list	2843
File selection	2843
Upload action	2846
Chunk upload	2848
Remove action	2850
Buttons	2851
Drag and drop	2851
Common	2851

RadioButton

Label and size in EJ2 JavaScript Radio button control

This section explains the different sizes and labels.

Label

RadioButton caption can be defined by using the [label](#) property. This reduces the manual addition of label for RadioButton. You can customize the label position before or after the RadioButton through the [labelPosition](#) property.

INDEX.TS

```
import { RadioButton } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Label position - Left.
let radiobutton: RadioButton = new RadioButton({ label: 'Left Side Label',
name: 'position', labelPosition: 'Before' });
radiobutton.appendTo('#radiobutton1');
//Label position - Right.
radiobutton = new RadioButton({ label: 'Right Side Label', name: 'position',
checked: true });
radiobutton.appendTo('#radiobutton2');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 RadioButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <ul>
      <li><input type="radio" id="radiobutton1"></li>
      <li><input type="radio" id="radiobutton2"></li>
    </ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  top: 45%;
  left: 45%;
}
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}

```

Size

The different RadioButton sizes available are default and small. To reduce the size of the default RadioButton to small, set the [cssClass](#) property to `e-small`.

INDEX.TS

```

import { RadioButton } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Small RadioButton.
let radiobutton: RadioButton = new RadioButton({ label: 'Small', name:
'size', checked: true, cssClass: 'e-small' });
radiobutton.appendTo('#radiobutton1');
//Default RadioButton.
radiobutton = new RadioButton({ label: 'Default', name: 'size' });
radiobutton.appendTo('#radiobutton2');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 RadioButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul>
            <li><input type="radio" id="radiobutton1"></li>
            <li><input type="radio" id="radiobutton2"></li>
        </ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
.e-radio-wrapper {
    margin-top: 18px;
}
li {
    list-style: none;
}
```

See Also

- [How to customize the RadioButton appearance](#)

Accessibility in EJ2 JavaScript Radio button control

The Radio button component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Radio button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Radio button component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Radio button component:

| Attributes | Purpose |

| --- | --- |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Radio button component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Radio button component.

| **Press** | **To do this** |

| --- | --- |

| **UP/Left arrow** | Move and select the previous options. |

| **Down/Right arrow** | Move and select the next options. |

Ensuring accessibility

The Radio button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Radio button component is shown in the following sample. Open the [sample](https://ej2.syncfusion.com/accessibility/Radio button.html) in a new window to evaluate the accessibility of the Radio button component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Customize radiobutton appearance in EJ2 JavaScript Radio button control

You can customize the appearance of the RadioButton component by using the CSS rules. Define own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

The background and border color of the RadioButton is customized through the custom classes to create the primary, success, info, warning, and danger type of RadioButton.

INDEX.TS

```
import { RadioButton } from '@syncfusion/ej2-buttons';
// To customize RadioButton appearance
// Refer the 'e-primary' class details in 'style.css'.
let radiobutton: RadioButton = new RadioButton({ label: 'Primary', name:
'custom', cssClass: 'e-primary' });
radiobutton.appendTo('#radiobutton1');
// Refer the 'e-success' class details in 'style.css'.
radiobutton = new RadioButton({ label: 'Success', name: 'custom', cssClass:
'e-success' });
radiobutton.appendTo('#radiobutton2');
// Refer the 'e-info' class details in 'style.css'.
radiobutton = new RadioButton({ label: 'Info', name: 'custom', cssClass: 'e-
info', checked: true });
radiobutton.appendTo('#radiobutton3');
// Refer the 'e-warning' class details in 'style.css'.
radiobutton = new RadioButton({ label: 'Warning', name: 'custom', cssClass:
'e-warning' });
radiobutton.appendTo('#radiobutton4');
```

```
// Refer the 'e-danger' class details in 'style.css'.
radiobutton = new RadioButton({ label: 'Danger', name: 'custom', cssClass:
'e-danger' });
radiobutton.appendTo('#radiobutton5');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 RadioButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <ul>
      <li><input type="radio" id="radiobutton1"></li>
      <li><input type="radio" id="radiobutton2"></li>
      <li><input type="radio" id="radiobutton3"></li>
      <li><input type="radio" id="radiobutton4"></li>
      <li><input type="radio" id="radiobutton5"></li>
    </ul>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  top: 45%;
```

```

left: 45%;
}
li {
list-style: none;
}
.e-radio-wrapper {
margin-top: 18px;
}
.e-success .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
background-color: #689f38;
border-color: #689f38;
}
.e-success .e-radio:checked:focus + label::after, .e-success .e-
radio:checked + label:hover::after { /* csslint allow: adjoining-classes */
background-color: #449d44;
border-color: #449d44;
}
.e-success .e-radio:checked + label::before {
border-color: #689f38;
}
.e-success .e-radio:checked:focus + label::before, .e-success .e-
radio:checked + label:hover::before { /* csslint allow: adjoining-classes */
border-color: #449d44;
}
.e-success .e-radio + label:hover::before {
border-color: #blafaf
}
.e-info .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
background-color: #2196f3;
border-color: #2196f3;
}
.e-info .e-radio:checked:focus + label::after, .e-info .e-radio:checked +
label:hover::after { /* csslint allow: adjoining-classes */
background-color: #0b7dda;
border-color: #0b7dda;
}
.e-info .e-radio:checked + label::before {
border-color: #2196f3;
}
.e-info .e-radio:checked:focus + label::before, .e-info .e-radio:checked +
label:hover::before {
border-color: #0b7dda;
}
.e-info .e-radio + label:hover::before {
border-color: #blafaf
}
.e-warning .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
background-color: #ef6c00;
border-color: #ef6c00;
}
.e-warning .e-radio:checked:focus + label::after, .e-warning .e-
radio:checked + label:hover::after { /* csslint allow: adjoining-classes */
background-color: #cc5c00;
}

```

```

.e-radio:checked + .e-warning::before {
border-color: #ef6c00;
}
.e-warning .e-radio:checked:focus + label::before, .e-warning .e-
radio:checked + label:hover::before {
border-color: #cc5c00;
}
.e-warning .e-radio + label:hover::before {
border-color: #blafaf
}
.e-danger .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
background-color: #d84315;
border-color: #d84315;
}
.e-danger .e-radio:checked:focus + label::after, .e-danger .e-radio:checked
+ label:hover::after { /* csslint allow: adjoining-classes */
background-color: #ba330a;
border-color: #ba330a;
}
.e-danger .e-radio:checked + label::before {
border-color: #d84315;
}
.e-danger .e-radio:checked:focus + label::before, .e-danger .e-radio:checked
+ label:hover::before {
border-color: #ba330a;
}
.e-danger .e-radio + label:hover::before {
border-color: #blafaf
}

```

Name and value in form submit in EJ2 JavaScript Radio button control

The [name](#) attribute of the RadioButton is used to group RadioButton. When the RadioButton are grouped in form, the checked items [value](#) attribute will be post to server on form submit that can be retrieved through the name. The disabled RadioButton value will not be sent to the server on form submit.

In the following code snippet, Credit and Debit card is in checked state.

Now, the value that is in checked state will be sent on form submit.

INDEX.TS

```

import { Button, RadioButton } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Name and Value attribute in form submit.
let radiobutton: RadioButton = new RadioButton({label: 'Credit/Debit Card',
name: 'payment', value: 'credit/debit', checked: true});
radiobutton.appendTo('#radiobutton1');
radiobutton = new RadioButton({label: 'Net Banking', name: 'payment', value:
'netbanking'});
radiobutton.appendTo('#radiobutton2');
radiobutton = new RadioButton({label: 'Cash on Delivery', name: 'payment',
value: 'cashondelivery'});
radiobutton.appendTo('#radiobutton3');

```



```

radiobutton = new RadioButton({label: 'Others', name: 'payment', value:
'others'}});
radiobutton.appendTo('#radiobutton4');
let button: Button = new Button({ isPrimary: true });
button.appendTo('#btnElement');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 RadioButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <form>
      <ul>
        <li><input type="radio" id="radiobutton1"></li>
        <li><input type="radio" id="radiobutton2"></li>
        <li><input type="radio" id="radiobutton3"></li>
        <li><input type="radio" id="radiobutton4"></li>
        <li><button id="btnElement">Submit</button></li>
      </ul>
    </form>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
}

```

```

width: 30%;
position: absolute;
top: 45%;
left: 45%;
}
.e-radio-wrapper {
margin-top: 18px;
}
button {
margin: 20px 0 0 5px;
}
li {
list-style: none;
}

```

Right to left in EJ2 JavaScript Radio button control

RadioButton component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in RadioButton component.

INDEX.TS

```

import { RadioButton } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize RadioButton component.
let radiobutton: RadioButton = new RadioButton({ label: 'Option 1', name:
'default', enableRtl: true });
// Render initialized RadioButton.
radiobutton.appendTo('#element1');
radiobutton = new RadioButton({ label: 'Option 2', name: 'default',
enableRtl: true, checked: true });
radiobutton.appendTo('#element2');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 RadioButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <input type="radio" id="element1">
  <input type="radio" id="element2">
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}

```

Set the disabled state in EJ2 JavaScript Radio button control

RadioButton component can be enabled/disabled by giving [disabled](#) property. To disable RadioButton component, the `disabled` property can be set as `true`.

The following example illustrates how to disable a radio button and the selected one is displayed using [change](#) event.

INDEX.TS

```

import { RadioButton, ChangeEventArgs } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize RadioButton component.
let radiobutton: RadioButton = new RadioButton({ label: 'Option 1', name:
'default', checked: true, change: onChange });
// Render initialized RadioButton.
radiobutton.appendTo('#element1');
radiobutton = new RadioButton({ label: 'Option 2', name: 'default',
disabled: true, change: onChange });
radiobutton.appendTo('#element2');
radiobutton = new RadioButton({ label: 'Option 3', name: 'default', change:
onChange });

```

```

radiobutton.appendTo('#element3');
function onChange(args: ChangeEventArgs): void {
    document.getElementById('text').innerText = 'Selected : ' + this.label;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 RadioButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul>
            <li><div id="text">Selected: Option 1</div></li>
            <li><input type="radio" id="element1"></li>
            <li><input type="radio" id="element2"></li>
            <li><input type="radio" id="element3"></li>
        </ul>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

```

}
.e-radio-wrapper {
  margin-top: 18px;
}
li {
  list-style: none;
}
#text {
  font-size: 16px;
}

```

Ej1 api migration in EJ2 JavaScript Radio button control

This article describes the API migration process of RadioButton component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| RadioButton Label | **Property:** *text*

 \$("#radiobutton").ejRadioButton({
text: "RadioButton"
 }); | **Property:** *label*

 var radioButton = new ej.buttons.RadioButton({
label: "RadioButton"
});
radioButton.appendTo("#radiobutton"); |

| Checked state | **Property:** *checked*

 \$("#radiobutton").ejRadioButton({
checked: true
 }); | **Property:** *checked*

 var radioButton = new ej.buttons.RadioButton({
checked: true
});
radioButton.appendTo("#radiobutton"); |

| Adding custom class | **Property:** *cssClass*

 \$("#radiobutton").ejRadioButton({
cssClass: "custom-class"
 }); | **Property:** *cssClass*

 var radioButton = new ej.buttons.RadioButton({
cssClass: "custom-class"
});
radioButton.appendTo("#radiobutton"); |

| Disabled state | **Property:** *enabled*

 \$("#radiobutton").ejRadioButton({
enabled: false
 }); | **Property:** *disabled*

 var radioButton = new ej.buttons.RadioButton({
disabled: true
});
radioButton.appendTo("#radiobutton"); |

| State persistence | **Property:** *enablePersistence*

 \$("#radiobutton").ejRadioButton({
enablePersistence: true
 }); | **Property:** *enablePersistence*

 var radioButton = new ej.buttons.RadioButton({
enablePersistence: true
});
radioButton.appendTo("#radiobutton"); |

| RTL | **Property:** *enableRTL*

 \$("#radiobutton").ejRadioButton({
enableRTL: true,
text: "RadioButton"
 }); | **Property:** *enableRtl*

 var radioButton = new ej.buttons.RadioButton({
enableRtl: true,
label: "RadioButton"
});
radioButton.appendTo("#radiobutton"); |

| HTML Attributes | **Property:** *htmlAttributes*

 \$("#radiobutton").ejRadioButton({
htmlAttributes : { required:"required" },
text: "RadioButton"
}); | Not applicable |

| Id property | **Property:** *id*

 \$("#radiobutton").ejRadioButton({
id: "sync"
 }); | Not applicable |

| Prefix value of Id | **Property:** *idPrefix*

 \$("#radiobutton").ejRadioButton({
idPrefix: "ej"
 }); | Not applicable |

| Name attribute | **Property:** *name*

 \$("#radiobutton").ejRadioButton({
 name: "sports"
 }); | **Property:** *name*

 var radioButton = new ej.buttons.RadioButton({
 name: "sports"
 });
 radioButton.appendTo("#radiobutton"); |

| Value attribute | **Property:** *value*

 \$("#radiobutton").ejRadioButton({
 value: "football",
 name: "sports"
 }); | **Property:** *value*

 var radioButton = new ej.buttons.RadioButton({
 value: "football",
 name: "sports"
 });
 radioButton.appendTo("#radiobutton"); |

| Size | **Property:** *size*

 \$("#radiobutton").ejRadioButton({
 size: "small"
 }); | **Property:** *cssClass*

 var radioButton = new ej.buttons.RadioButton({
 cssClass: "e-small"
 });
 radioButton.appendTo("#radiobutton"); |

| Validation rules | **Property:** *validationRules*

 \$("#radiobutton").ejRadioButton({
 validationRules: { required: true }
 }); | Not applicable |

| Validation message | **Property:** *validationMessage*

 \$("#radiobutton").ejRadioButton({
 validationRules: { required: true },
 validationMessage: { required: "Required RadioButton value" }
 }); | Not applicable |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Destroy | **Method:** *destroy*

 \$("#radiobutton").ejRadioButton({
 text: "RadioButton"
 });
 var radioButton = \$("#radiobutton").data("ejRadioButton");
 radioButton.destroy() | **Method:** *destroy*

 var radioButton = new ej.buttons.RadioButton({
 label: "RadioButton"
 });
 radioButton.appendTo("#radiobutton");
 radioButton.destroy(); |

| Disable the RadioButton | **Method:** *disable*

 \$("#radiobutton").ejRadioButton({
 text: "RadioButton"
 });
 var radioButton = \$("#radiobutton").data("ejRadioButton");
 radioButton.disable() | Not applicable |

| Enable the RadioButton | **Method:** *enable*

 \$("#radiobutton").ejRadioButton({
 text: "RadioButton"
 });
 var radioButton = \$("#radiobutton").data("ejRadioButton");
 radioButton.enable() | Not applicable |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| BeforeChange Event | **Events:** *beforeChange*

 \$("#radiobutton").ejRadioButton({
 beforeChange: function(args) { / code block */ }
 }); | Not applicable |

| Change Event | **Events:** *change*

 \$("#radiobutton").ejRadioButton({
 change: function(args) { / code block / }
 }); | **Events:** *change*

 var radioButton = new ej.buttons.RadioButton({
 change: function(args) { / code block / }
 });
 radioButton.appendTo("#radiobutton"); |

| Create Event | **Events:** *create*

 \$("#radiobutton").ejRadioButton({
 create: function(args) { / code block / }
 }); | **Events:** *created*

 var radioButton = new ej.buttons.RadioButton({
 created: function() { / code block / }
 });
 radioButton.appendTo("#radiobutton"); |

| Destroy Event | **Events:** *destroy*

 \$("#radiobutton").ejRadioButton({
destroy:
function(args) { / code block */ }
 }); | Not applicable |

Range Navigator

Getting started in EJ2 JavaScript Range navigator control

This section explains you the steps required to create a simple range navigator and demonstrate the basic usage of the range navigator control.

Dependencies

The list of minimum dependencies required to use a range navigator are follows:

```
`javascript
|-- @syncfusion/ej2-charts
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-calendars
\`
```

Setup for Local Development

Clone the Essential JS 2 quickstart application project from [GitHub](https://github.com/syncfusion/quickstart), and install the necessary npm packages using the following command line scripts.

```
`javascript
git clone https://github.com/angular/quickstart.git quickstart
cd quickstart
npm install
\`
```

Configuring System JS

Syncfusion Chart packages have to be mapped in the `system.config.js` configuration file to render range navigator.

```
`javascript
System.config({
  paths: {
    'syncfusion:': './node_modules/@syncfusion/',
  },
  map: {
```

```

app: 'app',
//Syncfusion packages mapping
"@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
"@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
"@syncfusion/ej2-charts": "syncfusion:ej2-charts/dist/ej2-charts.umd.min.js",
"@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
"@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
"@syncfusion/ej2-pdf-export": "syncfusion:ej2-pdf-export/dist/ej2-pdf-export.umd.min.js",
"@syncfusion/ej2-file-utils": "syncfusion:ej2-file-utils/dist/ej2-file-utils.umd.min.js",
"@syncfusion/ej2-compression": "syncfusion:ej2-compression/dist/ej2-compression.umd.min.js",
"@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",
"@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
"@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
"@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
"@syncfusion/ej2-svg-base": "syncfusion:ej2-svg-base/dist/ej2-svg-base.umd.min.js",
"@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js" },
packages: {
'app': { main: 'app', defaultExtension: 'js' }
}
});
System.import('app');
`

```

Add Range Navigator to the Project

Add the HTML div element for Range Navigator into your `index.html`. [src/index.html]

```

`html
<!DOCTYPE html>
<html lang="en">
<head>
<title>EJ2 Animation</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Typescript UI Controls" />
<meta name="author" content="Syncfusion" />

```



```

<link href="index.css" rel="stylesheet" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></script>
<script src="systemjs.config.js"></script>
</head>
<body>
<!--container which is going to render the RangeNavigator-->
<div id='element'>
</div>
</body>
</html>
`

```

Now import the RangeNavigator component into your `app.ts` to instantiate a RangeNavigator and append the range navigator instance to the `#element` [src/app/app.ts].

```

`ts
import { RangeNavigator } from '@syncfusion/ej2-charts';
// initialize RangeNavigator component
let range: RangeNavigator = new RangeNavigator();
// render initialized RangeNavigator
range.appendTo('#element');
`

```

Run the Application

The quickstart project is configured to compile and run the application in the browser. Use the following command to run the application.

```

npm start
`

```

The below example shows a basic Range Navigator.

INDEX.TS

```

import { RangeNavigator } from "@syncfusion/ej2-charts";
let range: RangeNavigator = new RangeNavigator();
range.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Module Injection

To create range navigator with additional features, inject the required modules. The following modules are used to extend rangenavigator's basic functionality.

- **AreaSeriesService** - Inject this module to use area series.
- **DateTimeService** - Inject this module to use date time axis.
- **RangeTooltipService** - Inject this module to show the tooltip.

Now import the above mentioned modules from chart package and inject it into the RangeNavigator component using **RangeNavigator.Inject** method.

```
`javascript
```

```
import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from '@syncfusion/ej2-charts';
```

```
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
```

```
,
```

Populate Range Navigator with Data

Now, we are going to provide data to the range navigator. Add a series object to the range navigator by using **series** property. Now map the field names x and y in the JSON data to the **xName** and **yName** properties of the series, then set the JSON data to **dataSource** property.

Since the JSON contains Datetime data, set the **valueType** as **DateTime**. By default, the axis **valueType** is **Numeric**.

INDEX.TS

```
import { RangeNavigator, AreaSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  labelFormat: 'MMM-yy',
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area',
  }],
  width: 2,
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: Get data from [here](#).

The sample should look like our [default](#), don't worry about the gradient color, let it takes the default color.

Enable Tooltip

The tooltip is useful to show the selected data. You can enable tooltip by setting the enable property as true in tooltip object and by injecting RangeTooltipService module using RangeNavigator.Inject(RangeTooltip) method.

INDEX.TS

```
import { RangeNavigator, AreaSeries, DateTime, RangeTooltip} from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  tooltip: { enable: true, displayMode: 'Always' },
  labelFormat: 'MMM-yy',
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area',
width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Selecting range in EJ2 JavaScript Range navigator control

The Range Selector's left and right thumbs are used to indicate the selected range in the large collection of data. A range can be selected in the following ways:

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and the end through the `value` property.

INDEX.TS

```
import { RangeNavigator, AreaSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime);
import { datasrc } from "./datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  labelFormat: 'MMM-yy',
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Lightweight in EJ2 JavaScript Range navigator control

By default, when the `dataSource` for `series` is empty, a lightweight Range Selector will be shown without Chart.

INDEX.TS

```
import { RangeNavigator, DateTime } from '@syncfusion/ej2-charts';
import { GetDateTimeData } from './datasource.ts';
RangeNavigator.Inject(DateTime);
```

```
let range: RangeNavigator = new RangeNavigator(
{
    valueType: 'DateTime',
    intervalType: 'Months',
    labelFormat: 'MMM',
    value: [new Date(2018, 4, 1), new Date(2018, 8, 1)],
    dataSource: GetDateTimeData(new Date(2018, 0, 1), new Date(2019, 0,
1)),
    xName: 'x', yName: 'y'
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Period Selector](#)

Series types in EJ2 JavaScript Range navigator control

To render the data, the Range Selector supports three types of series.

<!-- markdownlint-disable MD036 -->

Line

<!-- markdownlint-disable MD036 -->

To render a line series, use series type as **Line** and inject the **LineSeries** module using the **RangeNavigator.Inject(LineSeries)** method. By default, the line series is rendered in the Range Selector.

INDEX.TS

```
import { RangeNavigator, DateTime, LineSeries } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(DateTime, LineSeries);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  labelFormat: 'MMM-yy',
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Line', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Area

To render an area series, use series type as **Area** and inject **AreaSeries** module using **RangeNavigator.Inject(AreaSeries)** method.

INDEX.TS

```
import { RangeNavigator, DateTime, AreaSeries } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(DateTime, AreaSeries);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  labelFormat: 'MMM-yy',
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

StepLine

To render a Step line series, use series type as **Step Line** and inject **StepLineSeries** module using **RangeNavigator.Inject(StepLineSeries)** method.

INDEX.TS

```
import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
import { double } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
```



```

    value: [12, 30],
    series: [{
      dataSource: double,
      xName: 'x', yName: 'y', type: 'StepLine', width: 2,
    }],
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Data in EJ2 JavaScript Range navigator control

Numeric

The numeric scale is used to represent the numeric values of data in a Range Selector. By default, the **valueType** of a Range Selector is **Double**.

INDEX.TS

```

import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-
charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
import { double } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  value: [12, 30],
  series: [{
    dataSource: double,

```

```

      xName: 'x', yName: 'y', type: 'StepLine', width: 2,
    }],
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Range

The minimum and the maximum of the scale will be calculated automatically based on the provided data. It can be customized by using the `minimum`, `maximum`, and `interval` properties.

INDEX.TS

```

import { RangeNavigator, StepLineSeries } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries);
let range: RangeNavigator = new RangeNavigator({
  minimum: 10, maximum: 160, interval: 10,
  value: [60, 100],
  series: [{
    dataSource: [
      { xData: 10, yData: 35 }, { xData: 20, yData: 28 },
      { xData: 30, yData: 34 }, { xData: 40, yData: 32 },
      { xData: 50, yData: 40 }, { xData: 60, yData: 30 },
      { xData: 70, yData: 4 }, { xData: 80, yData: 22 },
      { xData: 90, yData: 30 }, { xData: 100, yData: 43 },
      { xData: 110, yData: 60 }, { xData: 120, yData: 33 },
      { xData: 130, yData: 40 }, { xData: 140, yData: 29 },
    ]
  }]
});

```

```

    { xData: 150, yData: 10 }, { xData: 160, yData: 16 },
  ],
  xName: 'xData', yName: 'yData', type: 'StepLine', width: 2,
}],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Format

The numeric labels can be formatted using the `labelFormat` property and it supports all the globalized formats.

INDEX.TS

```

import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-
charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
import { double } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  labelFormat: 'n1',
  value: [12, 30],
  series: [{
    dataSource: double,
    xName: 'x', yName: 'y', type: 'StepLine', width: 2,
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	\$1,000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place

1000	c2	\$1,000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place
------	----	------------	--

Custom Label Format

The Range Selector also supports the Custom Label formats using the placeholders such as **{value}\$**, in which the value represents the axis label, e.g. 20\$.

INDEX.TS

```
import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
import { double } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  // Custom label format
  labelFormat: '{value}$',
  value: [12, 30],
  series: [{
    dataSource: double,
    xName: 'x', yName: 'y', type: 'StepLine', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Logarithmic Axis

<!-- markdownlint-disable MD033 -->

The Logarithmic supports the logarithmic scale, and it is used to visualize the data when the Range Selector has numerical values in both the lower (e.g.: 10-6) and the higher (e.g.: 106) orders of the magnitude.

INDEX.TS

```
import { RangeNavigator, StepLineSeries, Logarithmic } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries, Logarithmic);
let data: object[] = [];
let max: number = 100;
for (let i: number = 0; i < 100; i++) {
    data.push({
        x: Math.pow(10, i * 0.1),
        y: Math.floor(Math.random() * (80 - 30 + 1)) + 30
    });
}
let range: RangeNavigator = new RangeNavigator({
    valueType: 'Logarithmic',
    value: [4, 6],
    interval: 1,
    series: [{
        dataSource: data, xName: 'x', yName: 'y', type: 'StepLine', width:
2,
    }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To use logarithmic scale, inject the Logarithmic module using the `RangeNavigator.Inject(Logarithmic)` method, and then set the `valueType` to `Logarithmic`.

Range

The minimum and the maximum of the Range Selector will be calculated automatically based on the provided data. It can be customized by using the `minimum`, `maximum`, and `interval` properties.

INDEX.TS

```

import { RangeNavigator, StepLineSeries, Logarithmic } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries, Logarithmic);
let data: object[] = [];
let max: number = 100;
for (let i: number = 0; i < 100; i++) {
    data.push({
        x: Math.pow(10, i * 0.1),
        y: Math.floor(Math.random() * (80 - 30 + 1)) + 30
    });
}
let range: RangeNavigator = new RangeNavigator({
    valueType: 'Logarithmic',
    value: [4, 6],
    interval: 1,
    series: [{
        dataSource: data, xName: 'x', yName: 'y', type: 'StepLine', width:
2,
    }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Logarithmic Base

The Logarithmic Base can be customized using the `logBase` property. The default value of this property is **10**.

INDEX.TS

```
import { RangeNavigator, StepLineSeries, Logarithmic } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries, Logarithmic);
let data: object[] = [];
let max: number = 100;
for (let i: number = 0; i < 100; i++) {
    data.push({
        x: Math.pow(10, i * 0.1),
        y: Math.floor(Math.random() * (80 - 30 + 1)) + 30
    });
}
let range: RangeNavigator = new RangeNavigator({
    valueType: 'Logarithmic',
    value: [4, 6],
    // logBase for logarithmic scale
    logBase: 2,
    series: [{
        dataSource: data, xName: 'x', yName: 'y', type: 'StepLine', width:
2,
    }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```



```

    <div id="container" style="margin-top: 125px">
      <div id="element"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Date-time

The Range Selector supports the DateTime scale and displays the DateTime values as labels in the specified format.

INDEX.TS

```

import { RangeNavigator, AreaSeries, DateTime } from "@syncfusion/ej2-
charts";
RangeNavigator.Inject(AreaSeries, DateTime);
import { datasrc } from "./datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date("2017-08-13"), new Date("2017-12-28")],
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Date-time Range navigator supports date-time scale and displays date-time values as a labels in the specified format.

Interval Customization

The DateTime intervals can be customized using the `interval` and the `intervalType` properties of the Range Selector. For example, if the `interval` is set to 2 and the `intervalType` is set to years, the interval will be considered to be 2 years.

DateTime supports the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes

INDEX.TS

```

import { RangeNavigator, AreaSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime);
import { datasrc } from "./datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  intervalType: 'Months',
  interval: 2,
  value: [new Date("2017-08-13"), new Date("2017-12-28")],
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Format

The `labelFormat` property is used to format and parse the date to all globalize format.

INDEX.TS

```

import { RangeNavigator, AreaSeries, DateTime } from "@syncfusion/ej2-
charts";
RangeNavigator.Inject(AreaSeries, DateTime);
import { datasrc } from "./datasource.ts";
let range: RangeNavigator = new RangeNavigator({
    valueType: 'DateTime',
    labelFormat: 'y/M/d',
    value: [new Date("2017-08-13"), new Date("2017-12-28")],
    series: [{
        dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
    }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container" style="margin-top: 125px">
      <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table shows the results of applying some common DateTime formats to the `labelFormat` property.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
new Date(2000, 03, 10)	EEEE	Monday	The Date is displayed in day format
new Date(2000, 03, 10)	yMd	04/10/2000	The Date is displayed in month/date/year format
new Date(2000, 03, 10)	MMM	Apr	The Shorthand month for the date is displayed
new Date(2000, 03, 10)	hm	12:00 AM	Time of the date value is displayed as label
new Date(2000, 03, 10)	hms	12:00:00 AM	The Label is displayed in hours:minutes:seconds format

Period selector in EJ2 JavaScript Range navigator control

The period selector allows to select a range with specified periods.

Periods

An array of objects that allows the users to specify pre-defined time intervals. The `interval` property specifies the count value of the button, and the `text` property specifies the text to be displayed on the button. The `intervaltype` property allows the users to customize the interval type, and it supports the following types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes

- Seconds

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
  <!--style reference from the Calendar component-->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 120px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

To use the period selector feature, inject the **PeriodSelector** module using the **RangeNavigator.Inject(PeriodSelector)** method.

Positioning period selector

The **position** property allows the users to position the period selector at the **Top** or **Bottom**.

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<!--style reference from the Calendar component-->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 120px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Height

The **height** property allows the users to specify the height of the period selector. The default value of the height property is **43px**.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
<!--style reference from the Calendar component-->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 120px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Visibility of range navigator

The `disableRangeSelector` property allows the users to display only the period selector and not the Range Selector.

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-charts/styles/material.css" rel="stylesheet">
    <!--style reference from the Calendar component-->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 120px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [LightWeight](#)

Labels in EJ2 JavaScript Range navigator control

Multilevel labels

The `labelIntersectAction` property is used to avoid overlapping of labels. The following code sample shows the setting of `labelIntersectAction` property to **Hide**.

INDEX.TS

```

import { RangeNavigator, DateTime, RangeTooltip } from '@syncfusion/ej2-
charts';
RangeNavigator.Inject(DateTime, RangeTooltip);
let data: object[] = [];
let value: number = 0; let point: object = {};
for (let j: number = 1; j < 1090; j++) {
    value += (Math.random() * 10 - 5);
    value = value < 0 ? Math.abs(value) : value;
    point = { x: new Date(2000, 0, j), y: value, z: value + 10 };
    data.push(point);
}
let range: RangeNavigator = new RangeNavigator(
{
    labelPosition: 'Outside',
    tooltip: { enable: true },
    valueType: 'DateTime',
    intervalType: 'Quarter',
    enableGrouping: true,
    value: [new Date(2001, 0), new Date(2002, 0)],
    dataSource: data,
    xName: 'x',
    yName: 'y'
}, '#element');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping

The second level axis labels can be grouped using “groupBy” property with the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

INDEX.TS

```

import { RangeNavigator, DateTime, RangeTooltip } from '@syncfusion/ej2-
charts';
RangeNavigator.Inject(DateTime, RangeTooltip);
let data: object[] = [];
let value: number = 0; let point: object = {};
for (let j: number = 1; j < 1090; j++) {
  value += (Math.random() * 10 - 5);

```

```

    value = value < 0 ? Math.abs(value) : value;
    point = { x: new Date(2000, 0, j), y: value, z: value + 10 };
    data.push(point);
}
let range: RangeNavigator = new RangeNavigator(
    {
        labelPosition: 'Outside',
        tooltip: { enable: true },
        valueType: 'DateTime',
        intervalType: 'Quarter',
        enableGrouping: true,
        groupBy: 'Years',
        value: [new Date(2001, 0), new Date(2002, 0)],
        dataSource: data,
        xName: 'x',
        yName: 'y'
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Smart labels

The `labelIntersectAction` property is used to avoid overlapping of labels. The following code sample shows the setting of `labelIntersectAction` property to **Hide**.

INDEX.TS

```
import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  labelIntersectAction: 'Hide',
  labelFormat: 'y/M/d',
  value: [new Date("2017-08-13"), new Date("2017-12-28")],
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'StepLine',
    width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Label positioning

By default, the labels can be placed outside the Range Selector. It can also be placed inside the Range Selector using the `labelPosition` property.

INDEX.TS

```
import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
```

```
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  labelPosition: 'Inside',
  value: [new Date("2017-08-13"), new Date("2017-12-28")],
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'StepLine',
    width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Labels customization

The font size, color, family, etc. can be customized using the `labelStyle` setting.

INDEX.TS

```
import { RangeNavigator, DateTime } from '@syncfusion/ej2-charts';
import { GetDateTimeData } from '../datasource.ts';
RangeNavigator.Inject(DateTime);
let range: RangeNavigator = new RangeNavigator(
  {
    valueType: 'DateTime',
    intervalType: 'Months',
    labelFormat: 'MMM',
```

```

        labelStyle: { color: 'red', size: '10px' },
        value: [new Date(2018, 5, 1), new Date(2018, 6, 1)],
        dataSource: GetDateTimeData(new Date(2018, 0, 1), new Date(2019, 0,
1)),
        xName: 'x', yName: 'y'
    }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid tick in EJ2 JavaScript Range navigator control

Grid line customization

The gridlines indicate axis divisions by drawing the chart plot. Gridlines include helpful cues to the user, particularly for large or complicated charts. The `width`, `color`, and `dashArray` of the major gridlines can be customized by using the `majorGridLines` setting.

INDEX.TS

```

import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-
charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
let range: RangeNavigator = new RangeNavigator({
    majorGridLines: {
        width: 4,
        color: 'blue',
        dashArray: '5,5'
    }
});

```

```

    },
    value: [25, 40],
    series: [{
      dataSource: [
        { xData: 10, yData: 35 }, { xData: 20, yData: 28 },
        { xData: 30, yData: 34 }, { xData: 40, yData: 32 },
        { xData: 50, yData: 40 }
      ],
      xName: 'xData', yName: 'yData', type: 'StepLine', width: 2,
    }],
  }, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tick line customization

Ticklines are the small lines which is drawn on the axis line representing the axis labels. Ticklines will be drawn outside the axis by default. The `width`, `color`, and `dashArray` of the major ticklines can be customized by using the `majorTickLines` setting.

INDEX.TS

```

import { RangeNavigator, StepLineSeries, DateTime } from "@syncfusion/ej2-
charts";
RangeNavigator.Inject(StepLineSeries, DateTime);
let range: RangeNavigator = new RangeNavigator({
  majorTickLines: {
    width: 3,

```

```

    color: 'red'
  },
  value: [25, 40],
  series: [{
    dataSource: [
      { xData: 10, yData: 35 }, { xData: 20, yData: 28 },
      { xData: 30, yData: 34 }, { xData: 40, yData: 32 },
      { xData: 50, yData: 40 }
    ],
    xName: 'xData', yName: 'yData', type: 'StepLine', width: 2,
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization in EJ2 JavaScript Range navigator control

Navigator appearance

The Range Selector can be customized by using the `navigatorStyleSettings`. The `selectedRegionColor` property is used to specify the color for the selected region, whereas the `unselectedRegionColor` property is used to specify the color for the unselected region.

INDEX.TS

```

import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);

```

```
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  tooltip: { enable: true },
  labelFormat: 'MMM-yy',
  navigatorStyleSettings: {
    unselectedRegionColor: 'skyblue',
    selectedRegionColor: 'pink'
  },
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
  '#element');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Thumb

The thumb property allows to customize the border, fill color, size, and type of thumb. Thumbs can be of two shapes: **Circle** and **Rectangle**.

INDEX.TS

```
import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);

```



```
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  tooltip: { enable: true },
  labelFormat: 'MMM-yy',
  navigatorStyleSettings: {
    thumb: {
      type: 'Rectangle',
      border: { width: 2, color: 'red' },
      fill: 'pink'
    }
  },
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Border customization

Using the `navigatorBorder`, the `width` and `color` of the Range Selector border can be customized.

INDEX.TS

```
import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  tooltip: { enable: true },
  labelFormat: 'MMM-yy',
  navigatorBorder: { width: 4, color: 'green' },
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Deferred update

If the `enableDeferredUpdate` property is set to **true**, then the changed event will be triggered after dragging the slider. If the `enableDeferredUpdate` is **false**, then the changed event will be triggered when dragging the slider. By default, the `enableDeferredUpdate` is set to **false**.

INDEX.TS

```
import { RangeNavigator, Chart, AreaSeries, DateTime, RangeTooltip,
IChangedEventArgs } from "@syncfusion/ej2-charts";
```

```

Chart.Inject(DateTime, AreaSeries);
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
import { datasrc } from "../datasource.ts";
let chart: Chart = new Chart(
    {
        primaryXAxis: {
            valueType: 'DateTime',
            edgeLabelPlacement: 'Shift'
        },
        series: [{
            dataSource: datasrc, xName: 'x', yName: 'y', width: 2, name:
'Rate', type: 'Area'
        }],
        tooltip: { enable: true },
        height: '350', legendSettings: { visible: false },
    }
);
chart.appendTo('#Chart');
let range: RangeNavigator = new RangeNavigator({
    valueType: 'DateTime',
    value: [new Date('2017-09-01'), new Date('2018-02-01')],
    tooltip: { enable: true },
    enableDeferredUpdate: true,
    series: [{
        dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
    }],
    changed: (args: IChangedEventArgs) => {
        chart.primaryXAxis.zoomFactor = args.zoomFactor;
        chart.primaryXAxis.zoomPosition = args.zoomPosition;
        chart.dataBind();
    },
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <div id="Chart"></div>
    </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Allow snapping

The `allowSnapping` property toggles the placement of the slider exactly to the left or on the nearest interval.

INDEX.TS

```
import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  tooltip: { enable: true },
  labelFormat: 'MMM-yy',
  allowSnapping: true,
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animation

The speed of the animation can be controlled using the `animationDuration` property. The default value of the `animationDuration` property is **500** milliseconds.

INDEX.TS

```

import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  tooltip: { enable: true },
  labelFormat: 'MMM-yy',
  animationDuration: 2000,
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Grid and Tick Lines](#)
- [Labels](#)

Tool tip in EJ2 JavaScript Range navigator control

<!-- markdownlint-disable MD036 -->

The tooltip for sliders are supported by the Range Selector. Sliders are used in the Range Selector to select data from a specific range. The tooltip displays the selected start and end values.

<!-- markdownlint-disable MD013 -->

Customization

Tooltip can be customized using the following properties:

- enable - Customizes the visibility of the tooltip.
- fill - Customizes the background color of the tooltip.
- opacity - Customizes the opacity of the tooltip.
- textStyle - Customizes the font size, color, family, style, weight, alignment, and overflow of the tooltip.

INDEX.TS

```
import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  tooltip: { enable: true, displayMode: 'Always', fill: 'red', opacity:
0.6, textStyle: { style: 'Italic', color: 'blue', size: '12px' } },
  labelFormat: 'MMM-yy',
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Label Format

The `labelFormat` property in the tooltip is used to format and parse the date to all globalize formats.

INDEX.TS

```

import { RangeNavigator, AreaSeries, DateTime, RangeTooltip } from
"@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
    valueType: 'DateTime',
    value: [new Date('2017-09-01'), new Date('2018-02-01')],
    tooltip: { enable: true, displayMode: 'Always', labelFormat: 'y/M/d' },
    series: [{
        dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
    }],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table shows the results of applying some common date and time formats to the `labelFormat` property.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
<code>new Date(2000, 03, 10)</code>	<code>EEEE</code>	Monday	The Date is displayed in day format
<code>new Date(2000, 03, 10)</code>	<code>yMd</code>	04/10/2000	The Date is displayed in month/date/year format
<code>new Date(2000, 03, 10)</code>	<code>MMM</code>	Apr	The Shorthand month for the date is displayed
<code>new Date(2000, 03, 10)</code>	<code>hm</code>	12:00 AM	Time of the date value is displayed as label
<code>new Date(2000, 03, 10)</code>	<code>hms</code>	12:00:00 AM	The Label is displayed in hours:minutes:seconds format

R t l in EJ2 JavaScript Range navigator control

The Range Selector supports right-to-left (RTL), which can be enabled with the `enableRtl` property.

INDEX.TS

```

import { RangeNavigator, AreaSeries, DateTime } from "@syncfusion/ej2-charts";
RangeNavigator.Inject(AreaSeries, DateTime);
import { datasrc } from "../datasource.ts";
let range: RangeNavigator = new RangeNavigator({
  valueType: 'DateTime',
  enableRtl: true,
  value: [new Date('2017-09-01'), new Date('2018-02-01')],
  labelFormat: 'MMM-yy',
  intervalType: 'Months',
  series: [{
    dataSource: datasrc, xName: 'x', yName: 'y', type: 'Area', width: 2,
  }],
}, '#element');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 125px">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Export print in EJ2 JavaScript Range navigator control

Export

The rendered Range Selector can be exported to **JPEG, PNG, SVG**, or **PDF** format by using the **export** method in the Range Selector. This method contains the following parameters:

- **Type** - To specify the export type. The component can be exported to **JPEG, PNG, SVG**, or **PDF** format.
- **File name** - To specify the file name to export.
- **Orientation** - To specify the orientation type. This is applicable only for PDF export type.

INDEX.TS

```

import { SplineSeries, AreaSeries, DateTime, Crosshair } from
 '@syncfusion/ej2-charts';
import { RangeNavigator, Chart, IChangedEventArgs } from '@syncfusion/ej2-
 charts';
import { RangeTooltip, Tooltip } from '@syncfusion/ej2-charts';
import { bitCoinData } from './datasource.ts';
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
let range: RangeNavigator = new RangeNavigator(
  {

```

```

        valueType: 'DateTime',
        value: [new Date(2017, 8, 1), new Date(2018, 1, 1)],
        tooltip: { enable: true },
        series: [
            {
                dataSource: bitCoinData, xName: 'x', yName: 'y', type:
'Area',
                width: 2, animation: { enable: false }
            },
        ],
    },
    ];
range.appendTo('#element');
document.getElementById('export').onclick = () => {
    range.export('PNG', 'result', null, [range]);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
        <div id="chart"></div>
        <button id="export" type="button" width="15%" style="float:
right">Export</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print

The rendered Range Selector can be printed directly from the browser by calling the public method `print`.

INDEX.TS

```

import { SplineSeries, AreaSeries, DateTime, Crosshair } from
 '@syncfusion/ej2-charts';
import { RangeNavigator, Chart, IChangedEventArgs } from '@syncfusion/ej2-
charts';
import { RangeTooltip, Tooltip } from '@syncfusion/ej2-charts';
import { bitCoinData } from './datasource.ts';
RangeNavigator.Inject(AreaSeries, DateTime, RangeTooltip);
let range: RangeNavigator = new RangeNavigator(
    {
        valueType: 'DateTime',
        value: [new Date(2017, 8, 1), new Date(2018, 1, 1)],
        tooltip: { enable: true },
        series: [
            {
                dataSource: bitCoinData, xName: 'x', yName: 'y', type:
'Area',
                width: 2, animation: { enable: false }
            },
        ],
    }
);
range.appendTo('#element');
document.getElementById('print').onclick = () => {
    range.print();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 125px">
        <div id="element"></div>
        <div id="chart"></div>
        <button id="print" type="button" width="15%" style="float:
right">Print</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Accessibility in EJ2 JavaScript Range navigator control

The Range navigator control followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Range navigator control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

WAI-ARIA attributes

The Range navigator control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Range navigator control:

- region (role)
- aria-label (attribute)

Keyboard interaction

The Range navigator control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Range navigator control.

| Press | To do this |

| --- | --- |

| Tab | Moves the focus to the Range navigator element. |

| Ctrl + P | Prints the Range navigator. |

Ensuring accessibility

The Range navigator control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Range navigator control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Range navigator control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

Ej1 api migration in EJ2 JavaScript Range navigator control

This article describes the API migration process of Chart component from Essential JS 1 to Essential JS 2.

RangeNavigator

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Allow snapping	Property:allowSnapping\$("#range").ejRangeNavigator({ allowSnapping: true;});	Property:allowSnappinglet range: RangeNavigator = new RangeNavigator({ allowSnapping: true;});range.appendTo('#element');

Animation duration	Not Applicable	Property:animationDuration let range: RangeNavigator = new RangeNavigator({ animationDuration: 2000});range.appendTo('#element');
Allow snapping	Property:allowSnapping \$("#range").ejRangeNavigator({ allowSnapping: true;});	Property:allowSnapping let range: RangeNavigator = new RangeNavigator({ allowSnapping: true});range.appendTo('#element');
Border for range navigator	Property:allowSnapping \$("#range").ejRangeNavigator({ border: { color: 'red', width: 2, opacity: 0.5 }});	Property:navigatorBorder let range: RangeNavigator = new RangeNavigator({ navigatorBorder:{ color: 'pink', width: 2});range.appendTo('#element');
dataSource for range navigator	Property:dataSource \$("#range").ejRangeNavigator({ dataSource: [{}]});	Property:dataSource let range: RangeNavigator = new RangeNavigator({ dataSource: [{}]});range.appendTo('#element');
enabling deferred update for range navigator	Property:enableDeferredUpdate \$("#range").ejRangeNavigator({ enableDeferredUpdate: true});	Property:enableDeferredUpdate let range: RangeNavigator = new RangeNavigator({ enableDeferredUpdate : false});range.appendTo('#element');
multilevel level labels	Property:labelSettings.higherLevelLabels \$("#range").ejRangeNavigator({ labelSettings.higherLevelLabels: { }});	Property:enableGrouping let range: RangeNavigator = new RangeNavigator({ enableGrouping : false});range.appendTo('#element');
enabling scroll bar	Property:enableScrollBar \$("#range").ejRangeNavigator({ enableScrollBar: true});	Not Applicable
enabling auto resizing	Property:enableAutoResize \$("#range").ejRangeNavigator({ enableScrollBar: true});	Not Applicable
responsive of range	Property:isResponsive \$("#range").ejRangeNavigator({ isResponsive: true});	Not Applicable

navigator		
enabling RTL for range navigator	Property:enableRtl\$("#range").ejRangeNavigator({ enableRtl: true});	Property:enableRtllet range: RangeNavigator = new RangeNavigator({ enableRtl : false});range.appendTo('#element');
interval for range navigator	Property:valueAxisSettings.range.interval\$("#range").ejRangeNavigator({ valueAxisSettings.range.interval = 1});	Property:intervallet range: RangeNavigator = new RangeNavigator({ interval : 1});range.appendTo('#element');
interval type for range navigator	Property:valueAxisSettings.range.interval\$("#range").ejRangeNavigator({ valueAxisSettings.intervalType = 'Years'});	Property:intervalTypelet range: RangeNavigator = new RangeNavigator({ intervalType : 'Months'});range.appendTo('#element');
label format for range navigator	Not applicable	Property:labelFormatlet range: RangeNavigator = new RangeNavigator({ labelFormat : 'yMd'});range.appendTo('#element');
label intersect action for range navigator	Not applicable	Property:labelIntersectActionlet range: RangeNavigator = new RangeNavigator({ labelIntersectAction : 'Hide'});range.appendTo('#element');
label style range navigator	Property:valueAxisSettings.font\$("#range").ejRangeNavigator({ valueAxisSettings.font: {}});	Property:labelStylelet range: RangeNavigator = new RangeNavigator({ labelStyle : 'Hide'});range.appendTo('#element');
locale of range navigator	Property:locale\$("#range").ejRangeNavigator({ locale: 'en-US'});	Property:labelStylelet range: RangeNavigator = new RangeNavigator({ locale: 'en-US'});range.appendTo('#element');
major grid lines of	Property:valueAxisSettings.majorGridLines\$("#range").ejRangeNavigator({ valueAxisSettings:	Property:majorGridLineslet range: RangeNavigator = new RangeNavigator({

range navigator	<pre>{ majorGridLines: { width: 2, color: 'red' } } };</pre>	<pre>majorGridLines: { width: 2, color: 'red' } } };range.appendTo('#element');</pre>
margin of range navigator	Not Applicable	Property:margin let range: RangeNavigator = new RangeNavigator({ margin: { } });range.appendTo('#element');
maximum value of range navigator	Property:valueAxisSettings.range.max \$("#range").ejRangeNavigator({ valueAxisSettings: { range: { max: 34 } } });	Property:maximum let range: RangeNavigator = new RangeNavigator({ maximum: 34 });range.appendTo('#element');
minimum value of range navigator	Property:valueAxisSettings.range.min \$("#range").ejRangeNavigator({ valueAxisSettings: { range: { min: 34 } } });	Property:minimum let range: RangeNavigator = new RangeNavigator({ minimum: 4 });range.appendTo('#element');
query for data source of range navigator	Not Applicable	Property:query let range: RangeNavigator = new RangeNavigator({ query: '' });range.appendTo('#element');
Secondary label alignment of range navigator	Not Applicable	Property:secondaryLabelAlignment let range: RangeNavigator = new RangeNavigator({ secondaryLabelAlignment: 'Far' });range.appendTo('#element');
Skeleton of range navigator axis	Not Applicable	Property:skeleton let range: RangeNavigator = new RangeNavigator({ skeleton: '' });range.appendTo('#element');
skeleton type of range navigator axis	Not Applicable	Property:skeletonType let range: RangeNavigator = new RangeNavigator({ skeletonType: '' });range.appendTo('#element');
Theme of range	Property:theme \$("#range").ejRangeNavigator({ theme: '' });	Property:theme let range: RangeNavigator = new RangeNavigator({ theme:

navigator		'});range.appendTo('#element');
Default selector value range navigator	Property:selectedRangeSettings\$("#range").ejRangeNavigator({ selectedRangeSettings: { start: 2, end: 20 } });	Property:valuelet range: RangeNavigator = new RangeNavigator({ value: [2, 10]});range.appendTo('#element');
Value type of range navigator	Property:valueType\$("#range").ejRangeNavigator({ valueType: 'DateTime' });	Property:valueTypelet range: RangeNavigator = new RangeNavigator({ valueType: 'Logarithmic'});range.appendTo('#element');
Width of range navigator	Property:size.width\$("#range").ejRangeNavigator({ size: { width: '200' } });	Property:widthlet range: RangeNavigator = new RangeNavigator({ width: '400'});range.appendTo('#element');
Height of range navigator	Property:size.height\$("#range").ejRangeNavigator({ size: { height: '200' } });	Property:heightlet range: RangeNavigator = new RangeNavigator({ height: '400'});range.appendTo('#element');
Series settings for range navigator	Property:seriesSettings\$("#range").ejRangeNavigator({ seriesSettings: { } });	Not Applicable
Range settings for range navigator	Property:rangeSettings\$("#range").ejRangeNavigator({ rangeSettings: { start: 3, end: 6 } });	Not Applicable
Scroll range settings for range navigator	Property:scrollRangeSettings\$("#range").ejRangeNavigator({ scrollRangeSettings: { start: 3, end: 6 } });	Not Applicable

Series

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
animation	Property:enableAnimation\$("#range").ejRangeNavigator({ enableAnimation: true});	Property:animationlet range: RangeNavigator = new RangeNavigator({ animation: { enable: true, duration: 2000 }});range.appendTo('#element');
Border for range navigator series	Not Applicable	Property:border{% raw %}let range: RangeNavigator = new RangeNavigator({ series: [{border:{ color: 'pink', width: 2}}]});range.appendTo('#element');{% endraw %}
dataSource for range navigator	Property:series.dataSource\$("#range").ejRangeNavigator({ series: [{ dataSource: [{}]}]});	Property:series.dataSourcelet range: RangeNavigator = new RangeNavigator({ series: [{ dataSource: []}]});range.appendTo('#element');
query for data source of range navigator	Not Applicable	Property:querylet range: RangeNavigator = new RangeNavigator({ series: [{ query: ''}]});range.appendTo('#element');
series type for range navigator	Property:series.type\$("#range").ejRangeNavigator({ series: [{ type: ''}]});	Property:series.typelet range: RangeNavigator = new RangeNavigator({ series: [{type: ''}]});range.appendTo('#element');
series xName for range navigator	Property:series.xName\$("#range").ejRangeNavigator({ series: [{ xName: ''}]});	Property:series.xNamelet range: RangeNavigator = new RangeNavigator({ series: [{ xName: ''}]});range.appendTo('#element');
series yName for range navigator	Property:series.yName\$("#range").ejRangeNavigator({ series: [{ yName: ''}]});	Property:series.yNamelet range: RangeNavigator = new RangeNavigator({ series: [{ yName: ''}]});range.appendTo('#element');

series fill color for range navigator	Property:series.fill\$("#range").ejRangeNavigator({ series: [{ fill: ' ' }]});	Property:series.filllet range: RangeNavigator = new RangeNavigator({ series: [{ fill: ' ' }]});range.appendTo('#element');
series width for range navigator	Property:series.width\$("#range").ejRangeNavigator({ series: [{ width: ' ' }]});	Property:series.widthlet range: RangeNavigator = new RangeNavigator({ series: [{ width: ' ' }]});range.appendTo('#element');
series dashArray for range navigator	Not Applicable	Property:series.dashArraylet range: RangeNavigator = new RangeNavigator({ series: [{ dashArray: ' ' }]});range.appendTo('#element');

StyleSettings

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Style settings of range navigator	Property:navigatorStyleSettings\$("#range").ejRangeNavigator({ navigatorStyleSettings: { } });	Property:styleSettingslet range: RangeNavigator = new RangeNavigator({ styleSettings: { } });range.appendTo('#element');
Selected region color of range navigator	Property:selectedRegionColor\$("#range").ejRangeNavigator({ navigatorStyleSettings: { selectedRegionColor: 'red' } });	Property:selectedRegionColorlet range: RangeNavigator = new RangeNavigator({ styleSettings: { selectedRegionColor: 'red' } });range.appendTo('#element');
UnSelected region color of range navigator	Property:unselectedRegionColor\$("#range").ejRangeNavigator({ navigatorStyleSettings: { unselectedRegionColor: 'red' } });	Property:unselectedRegionColorlet range: RangeNavigator = new RangeNavigator({ styleSettings: { unselectedRegionColor: 'red' } });range.appendTo('#element');

Thumb color of range navigator	Property:navigatorStyleSettings\$("#range").ejRangeNavigator({ navigatorStyleSettings: { thumbColor: 'red' } });	Property:thumbSettingslet range: RangeNavigator = new RangeNavigator({ styleSettings: { thumbSettings: 'pink' } });range.appendTo('#element');
Thumb color of range navigator	Property:navigatorStyleSettings\$("#range").ejRangeNavigator({ navigatorStyleSettings: { thumbColor: 'red' } });	Property:thumbSettingslet range: RangeNavigator = new RangeNavigator({ styleSettings: { thumbSettings: 'pink' } });range.appendTo('#element');
Selected region opacity of range navigator	Property:selectedRegionOpacity\$("#range").ejRangeNavigator({ navigatorStyleSettings: { selectedRegionOpacity: 0.4 } });	Not Applicable
Unselected region opacity of range navigator	Property:unselectedRegionOpacity\$("#range").ejRangeNavigator({ navigatorStyleSettings: { unselectedRegionOpacity: 0.4 } });	Not Applicable
Background for thumb	Property:background\$("#range").ejRangeNavigator({ navigatorStyleSettings: { background: 'red' } });	Not Applicable
border for thumb	Property:border\$("#range").ejRangeNavigator({ navigatorStyleSettings: { border: { color: 'red', width: 2 } } });	Not Applicable
Highlightsettings for range navigator	Property:highlightSettings\$("#range").ejRangeNavigator({ navigatorStyleSettings: { highlightSettings: { } } });	Not Applicable
Selected style settings for range navigator	Property:selectionSettings\$("#range").ejRangeNavigator({ navigatorStyleSettings: { selectionSettings: { } } });	Not Applicable
Left thumb template for range navigator	Property:leftThumbTemplate\$("#range").ejRangeNavigator({ navigatorStyleSettings: { leftThumbTemplate: '' } });	Not Applicable

Right thumb template for range navigator	Property:rightThumbTemplate\$("#range").ejRangeNavigator({ navigatorStyleSettings: { rightThumbTemplate: '' }});	Not Applicable
--	--	----------------

Tooltip

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
tooltip	Property:visible\$("#range").ejRangeNavigator({ tooltip: { visible: true }});	Property:enablelet range: RangeNavigator = new RangeNavigator({ tooltip: { enable: true }});range.appendTo('#element');
background color of tooltip	Property:backgroundColor\$("#range").ejRangeNavigator({ tooltip: { backgroundColor: 'red' }});	Property:filllet range: RangeNavigator = new RangeNavigator({ tooltip: { fill: 'pink' }});range.appendTo('#element');
Font style of tooltip	Property:font\$("#range").ejRangeNavigator({ tooltip: { font: 'red' }});	Property:textStylelet range: RangeNavigator = new RangeNavigator({ tooltip: { textStyle: 'pink' }});range.appendTo('#element');
Label format of tooltip	Property:labelFormat\$("#range").ejRangeNavigator({ tooltip: { labelFormat: 'yMd' }});	Property:formatlet range: RangeNavigator = new RangeNavigator({ tooltip: { format: 'yMd' }});range.appendTo('#element');
Display mode of tooltip	Property:tooltipDisplayMode\$("#range").ejRangeNavigator({ tooltip: { tooltipDisplayMode: 'always' }});	Property:displayModelet range: RangeNavigator = new RangeNavigator({ tooltip: { displayMode: 'Always' }});range.appendTo('#element');
Template of tooltip	Not Applicable	Property:templatelet range: RangeNavigator = new RangeNavigator({ tooltip: { template: 'Chart' }});

		' });range.appendTo('#element');
Border of tooltip	Not Applicable	Property: border let range: RangeNavigator = new RangeNavigator({ tooltip: { border: { color: 'red', width: 2 } }});range.appendTo('#element');
Opacity of tooltip	Not Applicable	Property: opacity let range: RangeNavigator = new RangeNavigator({ tooltip: { opacity: 0.5 } });range.appendTo('#element');

Period Selector

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
tooltip	Not Applicable	Property: enable let range: RangeNavigator = new RangeNavigator({ periodSelector: { periods: [{ interval: 1, intervalType: 'Months', text: '1M' }], position: 'Top' } });range.appendTo('#element');

Methods

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Print	Not Applicable	Property: print() let range: RangeNavigator = new RangeNavigator({});range.print('chart');
Export	Not Applicable	Property: export() let range: RangeNavigator = new RangeNavigator({});range.export('JPEG', 'chart');

Events

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Fires before loading the	Property: load \$("#rangeNavigator").ejRangeNavigator({ load: () { } });	Property: load let rangeNavigator: RangeNavigator = new RangeNavigator({ load: () => { } });rangeNavigator.appendTo('#rangeNavigator');

RangeNavigator.		
Fires before loading the RangeNavigator.	Property:loaded\$("#rangeNavigator").ejRangeNavigator({ loaded: () { }});	Property:loadedlet rangeNavigator: RangeNavigator = new RangeNavigator({ loaded: () => { }});rangeNavigator.appendTo('# rangeNavigator');
Fires when the value changes in range navigator.	Property:rangeChanged\$("#rangeNavigator").ejRangeNavigator({ rangeChanged: () { }});	Property:changedlet rangeNavigator: RangeNavigator = new RangeNavigator({ changed: () => { }});rangeNavigator.appendTo('# rangeNavigator');
Fires before after the RangeNavigator.	Not Applicable	Property:changedlet rangeNavigator: RangeNavigator = new RangeNavigator({ resized: () => { }});rangeNavigator.appendTo('# rangeNavigator');
Fires before tooltip the RangeNavigator.	Not Applicable	Property:tooltipRenderlet rangeNavigator: RangeNavigator = new RangeNavigator({ tooltipRender: () => { }});rangeNavigator.appendTo('# rangeNavigator');
Fires before period render in the RangeNavigator.	Not Applicable	Property:selectorRenderlet rangeNavigator: RangeNavigator = new RangeNavigator({ selectorRender: () => { }});rangeNavigator.appendTo('# rangeNavigator');
Fires when scrollStart the RangeNavigator.	Property:scrollStart\$("#rangeNavigator").ejRangeNavigator({ scrollStart: () { }});	Not Applicable
Fires when scrollEnd the	Property:scrollEnd\$("#rangeNavigator").ejRangeNavigator({ scrollEnd: () { }});	Not Applicable

RangeNavigator.		
Fires when selected range Start the RangeNavigator.	Property:selectedRangeStart\$("#rangeNavigator").ejRangeNavigator({selectedRangeStart: () { }});	Not Applicable
Fires when selected range ends the RangeNavigator.	Property:selectedRangeEnd\$("#rangeNavigator").ejRangeNavigator({selectedRangeEnd: () { }});	Not Applicable
Fires when scroll range changed in the RangeNavigator.	Property:scrollChanged\$("#rangeNavigator").ejRangeNavigator({ scrollChanged: () { }});	Not Applicable
Fires when click in the RangeNavigator.	Property:click\$("#rangeNavigator").ejRangeNavigator({ click: () { }});	Not Applicable
Fires when right click in the RangeNavigator.	Property:rightClick\$("#rangeNavigator").ejRangeNavigator({ rightClick: () { }});	Not Applicable
Fires when double click in the RangeNavigator.	Property:doubleClick\$("#rangeNavigator").ejRangeNavigator({ doubleClick: () { }});	Not Applicable

Range Slider

Getting started in EJ2 JavaScript Range slider control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

control Initialization

The Essential JS 2 JavaScript controls can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder `myapp` for Essential JS 2 JavaScript controls.

Step 2: You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

Syntax:

Script: `(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\dist\global\{PACKAGE_NAME}.min.js`

Styles: `(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\styles\material.css`

Example:

Script: `C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-inputs\dist\global\ej2-inputs.min.js`

Styles: `C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-inputs\styles\material.css`

The below located script and style file contains all Syncfusion JavaScript (ES5) UI control resources in a single file.

Scripts: `(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\dist\ej2.min.js`

Styles: `(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\material.css`

Step 3: Create a folder `myapp/resources` and copy/paste the global scripts and styles from the above installed location to `myapp/resources` location.

Step 4: Create a HTML page (index.html) in `myapp` location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Inputs's global and dependent script -->
<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-inputs.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the **Slider** element and initiate the **Essential JS 2 Slider** control in the **index.html** by using following code

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Inputs's global and dependent script -->
<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-inputs.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element -->
<div id='element'></div>
<script>
// Initialize Essential JS 2 JavaScript Slider control
```

```

var sliderInstance = new ej.inputs.Slider({ value: 30 });
//Render initialized Slider
sliderInstance.appendTo("#element");
</script>
</body>
</html>
`

```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Slider** control.

Using CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript controls.

Step 2: The Essential JS 2 control's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: `https://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `https://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <https://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js>

Styles: <https://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css>

Step 3: Create a HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `Slider` element and initiate the **Essential JS 2 Slider** control in the `index.html` by using following code.

INDEX.HTML

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>

    <title>Essential JS 2</title>
    <!-- Essential JS 2 material theme -->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet" type="text/css"/>
    <!-- Essential JS 2 Input's global script -->
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/dist/global/ej2-base.min.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/dist/global/ej2-buttons.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/dist/global/ej2-popups.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/dist/global/ej2-inputs.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!-- Add the HTML <div> element -->
  <div id='element'></div>
  <script>
    // Initialize Essential JS 2 JavaScript Slider control
    var sliderInstance = new ej.inputs.Slider({value: 30});
    //Render initialized Slider
    sliderInstance.appendTo("#element");
  </script>
</body>
</html>

```

INDEX.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  height: 140px;
  margin: 100px auto;
  padding: 30px 10px;
  width: 260px;
}

```

Step 4: Now, run the `index.html` in web browser, it will render the `Essential JS 2 Slider` control.

Need to refer dependency control styles and scripts as like above example. We can also use [CRG](#) to generate combined control styles.

See Also

[Slider Types](#)

[Slider Formatting](#)

[Orientation Slider](#)

[Ticks in Slider](#)[Tooltip in Slider](#)[Ticks in EJ2 JavaScript Range slider control](#)

The Ticks in Slider supports you to easily identify the current value/values of the Slider. It contains `smallStep` and `largeStep`. The value of the major ticks alone will be displayed in the slider. In order to enable/disable the small ticks, use the `showSmallTicks` property.

INDEX.TS

```
import { Slider } from '@syncfusion/ej2-inputs';
// Initialize Slider control
let sliderObj: Slider = new Slider({
    tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
    value: 30,
    // Slider ticks customization
    ticks: { placement: 'After', largeStep: 20, smallStep: 10,
showSmallTicks: true }
});
// Render initialized Slider
sliderObj.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <div id="slider">
        </div>
      </div>
    </div>
  </div>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 260px;
    margin: 0 auto;
    padding: 30px 10px;
    width: 260px;
}

```

Step

When the Slider is moved, it increases/decreases the value based on the step value. By default, the value is increased/decreased by 1. Use the `step` property to change the increment step value.

INDEX.TS

```

import { Slider } from '@syncfusion/ej2-inputs';
// Initialize Slider control
let rangeObj: Slider = new Slider({
    ticks: { placement: 'After', largeStep: 20, smallStep: 10,
showSmallTicks: true },
    tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
    value: 30,
    // Enables step
    step: 10
});
// Render initialized Slider
rangeObj.appendTo('#slider');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Slider</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Slider Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="slider">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 260px;
    margin: 0 auto;
    padding: 30px 10px;
}

```

```
width: 260px;
}
```

Min and Max

Enables the minimum/starting and maximum/ending value of the Slider, by using the `min` and `max` property. By default, the minimum value is 1 and maximum value is 100. In the following sample the slider is rendered with the min value as 100 and max value as 1000.

INDEX.TS

```
import { Slider } from '@syncfusion/ej2-inputs';
// Initialize Slider control
let rangeObj: Slider = new Slider({
    ticks: { placement: 'After', largeStep: 200, smallStep: 100,
showSmallTicks: true },
    tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
    // Minimum value
    min: 100,
    // Maximum value
    max: 1000,
    // Slider current value
    value: 400
});
// Render initialized Slider
rangeObj.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Slider</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Slider Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="slider">
```



```

        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c8f;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 260px;
    margin: 0 auto;
    padding: 30px 10px;
    width: 260px;
}

```

Format in EJ2 JavaScript Range slider control

The **format** feature used to customize the units of Slider values to desired format. The formatted values will also be applied to the ARIA attributes of the slider. There are two ways of achieving formatting in slider.

- Use the [format](#) API of slider which utilizes our [Internationalization](#) to format values.
- Customize using the events namely **renderingTicks** and **tooltipChange**.

INDEX.TS

```

import { Slider } from '@syncfusion/ej2-inputs';
// Initialize Slider control
let sliderObj: Slider = new Slider({
    min: 0, max: 100, step: 1, value: 30,
    // Applying currency formatting for tooltip with two decimal specifiers
    tooltip: { isVisible: true, format: 'C2' },
    // Applying currency formatting for ticks with two decimal specifiers
    ticks: { placement: 'After', format: 'C2', largeStep: 20, smallStep: 10,
    showSmallTicks: true }
});

```

```
});
// Render initialized Slider
sliderObj.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <div id="slider">
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

INDEX.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
```

```

position: absolute;
top: 45%;
width: 30%;
}
.wrap {
box-sizing: border-box;
height: 260px;
margin: 0 auto;
padding: 30px 10px;
width: 460px;
}

```

Using format API

In this method, we have different predefined formatting styles like Numeric (N), Percentage (P), Currency (C) and # specifiers. In this below example we have formatted the ticks and tooltip values into percentage.

INDEX.TS

```

import { Slider } from '@syncfusion/ej2-inputs';
// Initialize Slider control
let percentObj: Slider = new Slider({
  min: 0, max: 1, value: .3, step: .01,
  // Assigning ticks values to percentage formatting
  ticks: { placement: 'After', largeStep: .2, smallStep: .1,
  showSmallTicks: true, format: 'P0' },
  // Assigning tooltip values to percentage formatting
  tooltip: { placement: 'Before', isVisible: true, showOn: 'Always',
  format: 'P0' },
});
// Render initialized Slider
percentObj.appendTo('#slider');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="slider">
            </div>
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

INDEX.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 260px;
    margin: 0 auto;
    padding: 30px 10px;
    width: 460px;
}
```

Using Events

In this method, we will be retrieving the values from the slider events then process them to desired formatted the values. In this sample we have customized the **ticks** values into weekdays as one formatting and **tooltip** values into day of the week as another formatting.

INDEX.TS

```
import { Slider, SliderTickEventArgs, SliderTooltipEventArgs } from
'@syncfusion/ej2-inputs';
// Initialize Slider control
let eventObj: Slider = new Slider({
    // Minimum value
```

```

min: 0,
// Maximum value
max: 6,
// Slider current value
value: 2,
// Assigning ticks data
ticks: {
    placement: 'After',
    largeStep: 1
},
renderingTicks: function (args: SliderTickEventArgs) {
    // Weekdays Array
    let daysArr: string [] =
['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];
    // Customizing each ticks text into weekdays
    args.text = daysArr[parseFloat(args.value as any)];
},
tooltipChange: function (args: SliderTooltipEventArgs) {
    // Customizing tooltip to display the Day (in numeric) of
the week
    args.text = 'Day ' + (Number(args.value) + 1).toString();
},
// Assigning tooltip data
tooltip: {
    placement: 'Before',
    isVisible: true
}
});
// Render initialized Slider
eventObj.appendTo('#slider');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div class="wrap">
        <div id="slider">
        </div>
      </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  height: 260px;
  margin: 0 auto;
  padding: 30px 10px;
  width: 460px;
}

```

Limits in EJ2 JavaScript Range slider control

The slider limits restrict the slider thumb between a particular range. This is used if higher or lower value affects the process or product where the slider is being used.

The following are the six options in the slider's limits object. Each API in the limits object is optional.

- enabled: Enables the limits in the Slider.
- minStart: Sets the minimum limit for the first handle.
- minEnd: Sets the maximum limit for the first handle.
- maxStart: Sets the minimum limit for the second handle.
- maxEnd: Sets the maximum limit for the second handle.
- startHandleFixed: Locks the first handle.
- endHandleFixed: Locks the second handle.

Default and MinRange Slider limits

There is only one handle in the Default and MinRange Slider, so minStart, minEnd, and startHandleFixed options can be used.

When the limits are enabled in the Slider, the limited area becomes darken. So you can differentiate the allowed and restricted area.

Refer to the following snippet to enable the limits in the Slider.

```
`ts
let slider = new Slider({
.....
limits: { enabled: true, minStart: 10, minEnd: 40 }
.....
});
`
```

INDEX.TS

```
import { Slider } from '@syncfusion/ej2-inputs';
// Initialization of Slider
let slider: Slider = new Slider({
  min: 0,
  max: 100,
  value: 30,
  type: 'MinRange',
  limits: { enabled: true, minStart: 10, minEnd: 40 },
  tooltip: { isVisible: true }
});
// Render initialized Slider
slider.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="slider">
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 140px;
    margin: 100px auto;
    padding: 30px 10px;
    width: 260px;
}

```

Range Slider limits

In the range slider, both handles can be restricted and locked from the limit's object. In this sample, the first handle is limited between

10 and 40, and the second handle is limited between 60 and 90.

```
`ts
```

```
let slider = new Slider({
```

```
.....
```

```
limits: { enabled: true, minStart: 10, minEnd: 40, maxStart: 60, maxEnd: 90 }
```


.....

});

,

INDEX.TS

```
import { Slider } from '@syncfusion/ej2-inputs';
// Initialization of Slider
let slider: Slider = new Slider({
  min: 0,
  max: 100,
  type: 'Range',
  value: [30, 70],
  limits: { enabled: true, minStart: 10, minEnd: 40, maxStart: 60, maxEnd:
90 },
  tooltip: { isVisible: true }
});
// Render initialized Slider
slider.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <div id="slider">
        </div>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  height: 140px;
  margin: 100px auto;
  padding: 30px 10px;
  width: 260px;
}

```

Handle lock

The movement of slider handles can be locked by enabling the startHandleFixed and endHandleFixed properties in the limit's object. In this sample, the movement of both slider handles has been locked.

`ts

```

let slider = new Slider({
  .....
  limits: { enabled: true, startHandleFixed: true, endHandleFixed: true }
  .....
});
`

```

INDEX.TS

```

import { Slider } from '@syncfusion/ej2-inputs';
// Initialization of Slider
let slider: Slider = new Slider({
  min: 0,
  max: 100,
  type: 'Range',
  value: [30, 70],
  limits: { enabled: true, startHandleFixed: true, endHandleFixed: true },

```

```

    tooltip: { isVisible: true }
});
// Render initialized Slider
slider.appendTo('#slider');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <div id="slider">
        </div>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008c9f;
  font-family: 'Helvetica Neue', 'calibri';
  font-size: 14px;
  height: 40px;
}

```

```

left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
  box-sizing: border-box;
  height: 140px;
  margin: 100px auto;
  padding: 30px 10px;
  width: 260px;
}

```

Globalization and localization in EJ2 JavaScript Range slider control

Localization

The [Localization](#) library allows you to localize default text content of the Slider. The slider control has static text on some features (like increase and decrease button) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and its values are used in the slider.

Locale keywords | Text

Increase | Increase

Decrease | Decrease

Loading translations

To load translation object in an application, use [load](#) function of the [L10n](#) class.

The following example demonstrates the Slider in **Deutsch** culture.

INDEX.TS

```

import { Slider } from '@syncfusion/ej2-inputs';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'de-DE': {
    'slider': {
      incrementTitle: 'Erhöhen, ansteigen', decrementTitle:
'verringern'
    }
  }
});
// Initialize range Slider control
let rangeObj: Slider = new Slider({
  value: [30, 70],
  type: 'Range',
  locale: 'de-DE',
  showButtons: true
});
rangeObj.appendTo('#slider');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 Slider</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Slider Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="slider">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 260px;
}

```

```
margin: 0 auto;  
padding: 30px 10px;  
width: 260px;  
}
```

See Also

- [Internationalization](#)
- [Localization](#)

Style in EJ2 JavaScript Range slider control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the slider track

Use the following CSS to customize the slider track.

```
,  
  
.e-control-wrapper.e-slider-container.e-horizontal .e-slider-track {  
background: #007bff;  
height: 3px;  
}  
,
```

Customizing the slider handle

Use the following CSS to customize the slider handle properties.

```
,  
  
.e-control-wrapper.e-slider-container .e-slider .e-handle {  
background-color: #f9920b;  
border-radius: 50%;  
border: 0;  
}  
,
```

Customizing the slider limits

Use the following CSS to customize the slider limits.

```
,  
  
.e-control-wrapper.e-slider-container.e-horizontal .e-limits {  
background-color: rgba(69, 100, 233, 0.46);  
}  
,
```

Customizing the slider ticks

Use the following CSS to customize the slider ticks.

```
,
.e-scale .e-tick.e-custom::before {
content: '\e967';
position: absolute;
}
,
```

Customizing the slider buttons

Use the following CSS to customize the slider buttons.

```
,
.e-control-wrapper.e-slider-container .e-slider-button {
background: #007bff;
height: 25px;
width: 25px;
}
,
```

Accessibility in EJ2 JavaScript Range Slider component

The Range Slider component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Range Slider component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```
<style>
```

```
.post .post-content img {
```

```
display: inline-block;
```

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Range Slider component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Range Slider component:

| Attributes | Purpose |

| --- | --- |

| **role=slider** | Used to convey a significant and contextual message to the user. |

| **aria-valuemin** | Indicates the Minimum value of the slider. |

| **aria-valuemax** | Indicates the Maximum value of the slider. |

| **aria-valuenow** | Indicates the current value of the slider. |

| **aria-valuetext** | Returns the current text of the slider. |

| **aria-orientation** | Indicates whether the Slider is oriented horizontally or vertically. |

| **aria-label** | Provides an accessible name for the Slider, serving as label text for the Slider's left and right buttons (for increment and decrement). |

Keyboard interaction

The Range Slider component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Range Slider component.

| **Press** | **To do this** |

| --- | --- |

| Right Arrow/Up Arrow | Increase the Slider value. |

| Left Arrow/Down Arrow | Decrease the Slider value. |

| Home | Moves to the start value (for Range Slider when the second thumb is focused and the Home key is pressed, it moves to the first thumb value). |

| <kbd>End | Moves to the end value (for Range Slider when the first thumb is focused and the End key is pressed, it moves to the second thumb value). |

| Page Up | Increases the Slider by `largeStep` value. |

| Page Down | Decreases the Slider by `largeStep` value. |

Ensuring accessibility

The Range Slider component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Range Slider component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Range Slider component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

Ej1 api migration in EJ2 JavaScript Range slider control

This article describes the API migration process of Slider component from Essential JS 1 to Essential JS 2

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Max value | **Property:** `maxValue`
 `$("#slider").ejSlider({
 maxValue: 60
 });` | **Property:** `max`
 let slider: Slider = new ej.inputs.Slider({
 max : 100
 });
 slider.appendTo('#test'); |

| Min value | **Property:** `minValue`
 `$("#slider").ejSlider({
 minValue: 60
 });` | **Property:** `min`
 let slider: Slider = new ej.inputs.Slider({
 min : 10
 });
 slider.appendTo('#test'); |

| Step | **Property:** `incrementStep, largeStep, smallStep, showSmallTicks`
 `$("#slider").ejSlider({
 incrementStep: 20,
 smallStep: 5,
 largeStep: 40,
 showSmallTicks: true
 });` | **Property:** `ticks`
 let slider: Slider = new ej.inputs.Slider({
 ticks: { placement: 'After', largeStep: 20, smallStep: 10, showSmallTicks: true },
 });
 slider.appendTo('#test'); |

| Type | **Property:** `sliderType`
 `$("#slider").ejSlider({
 sliderType: ej.SliderType.Default
 });` | **Property:** `type`
 let slider: Slider = new ej.inputs.Slider({
 type: 'Range'
 });
 slider.appendTo('#test'); |

| Tooltip | **Property:** `showTooltip`
 `$("#slider").ejSlider({
 showTooltip: true
 });` | **Property:** `tooltip`
 let slider: Slider = new ej.inputs.Slider({
 tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
 });
 slider.appendTo('#test'); |

| RTL | **Property:** *enableRTL*
 \$("#slider").ejSlider({
 enableRTL: false
 }); | **Property:** *enableRtl*
 let slider: Slider = new ej.inputs.Slider({
 enableRtl: false
 });
 slider.appendTo('#test'); |

| Custom values | **Not Applicable** | **Property:** *customValues*
 let slider: Slider = new ej.inputs.Slider({
 customValues: ['Mon', 'Tue', 'Wed'],
});
 slider.appendTo('#test'); |

| Limit the slider movement | **Not Applicable** | **Property:** *limits*
 let slider: Slider = new ej.inputs.Slider({
 type: 'MinRange',
 limits: { enabled: true, minStart: 10, minEnd: 40 }
});
 slider.appendTo('#test'); |

| Disable | **Method:** *disable*
 var slider = \$("#slider").data("ejSlider");
 slider.disable(); | **Property:** *enabled*
 let slider: Slider = new ej.inputs.Slider({
 value: 30
});
 slider.appendTo('#test');
 slider.enabled = false;
 |

| Enable | **Method:** *enable*
 var slider = \$("#slider").data("ejSlider");
 slider.enable(); | **Property:** *enabled*
 let slider: Slider = new ej.inputs.Slider({
 value: 30,
 enabled: false
});
 slider.appendTo('#test');
 slider.enabled = true;
 |

| Set Value | **Method:** *setValue(value, [enableAnimation])*
 var slider = \$("#slider").data("ejSlider");
 slider.setValue(10); | **Property:** *value*
 let slider: Slider = new ej.inputs.Slider({ });
 slider.appendTo('#test');
 slider.value = 30;
 |

| Get Value | **Method:** *getValue()*
 var slider = \$("#slider").data("ejSlider");
 slider.getValue(); | **Property:** *value*
 let slider: Slider = new ej.inputs.Slider({
 Value: 30
});
 slider.appendTo('#test');
 slider.value;
 |

| Destroy | **Not Applicable** | **Method:** *destroy()*
 let slider: Slider = new ej.inputs.Slider({});
 slider.appendTo('#test');
 slider.destroy(); |

| Change | **Event:** *change*
 \$("#slider").ejSlider({ change: function (args) {} }); | **Event:** *changed*
 let slider: Slider = new ej.inputs.Slider({
 changed: function(e: Event): void { }
 });
 slider.appendTo('#test'); |

| Create | **Event:** *create*
 \$("#slider").ejSlider({ create: function (args) {} }); | **Event:** *created*
 let slider: Slider = new ej.inputs.Slider({
 created: function(e: Event): void { }
 });
 slider.appendTo('#test'); |

| Destroy | **Event:** *destroy*
 \$("#slider").ejSlider({ destroy: function (args) {} }); | **Method:** *destroy()*
 let slider: Slider = new ej.inputs.Slider({
 value: 30
});
 slider.appendTo('#test');
 slider.destroy();
 |

| Slide | **Event:** *slide*
 \$("#slider").ejSlider({ slide: function (args) {} }); | **Event:** *change*
 let slider: Slider = new ej.inputs.Slider({
 change: function(e: Event): void { }
 });
 slider.appendTo('#test'); |

| Start | **Event:** *start*
 \$("#slider").ejSlider({ start: function (args) {} }); | **Event:** *created*
 let slider: Slider = new ej.inputs.Slider({
 created: function(e: Event): void { }
 });
 slider.appendTo('#test'); |

| Stop | **Event:** *stop*
 \$("#slider").ejSlider({ stop: function (args) {} }); | **Event:** *changed*
 let slider: Slider = new ej.inputs.Slider({
 changed: function(e: Event): void { }
 });
 slider.appendTo('#test');
 |

| Rendered Ticks | **Not Applicable** | **Event:** *renderedTicks*
 let slider: Slider = new ej.inputs.Slider({
 renderedTicks: function(e: Event): void { }
 });
 slider.appendTo('#test'); |

How To

Format value using slider in EJ2 JavaScript Range slider control

Achieve date format

The date formatting can be achieved in **ticks** and **tooltip** using **renderingTicks** and **tooltipChange** events, respectively. The process of formatting is explained in the following sample.

INDEX.TS

```
import { Slider, SliderTickEventArgs, SliderTooltipEventArgs } from
 '@syncfusion/ej2-inputs';
// Initialize slider control
let dateObj: Slider = new Slider({
  /**
   * Initialize the min and max values of the slider using JavaScript date
   functions
   * new Date (Year, Month, day, hours, minutes, seconds, milliseconds)
   */
  min: new Date("2013-06-13").getTime(),
  value: new Date("2013-06-15").getTime(),
  max: new Date("2013-06-21").getTime(),
  // 86400000 milliseconds for a day
  step: 86400000,
  tooltipChange: function (args: SliderTooltipEventArgs) {
    let totalMiliSeconds = Number(args.text);
    // Convert the current milliseconds to the respective date in
    desired format
    let custom = { year: "numeric", month: "short", day: "numeric" };
    args.text = new Date(totalMiliSeconds).toLocaleDateString("en-us",
    custom);
  },
  tooltip: {
    placement: 'Before',
    isVisible: true
  },
  renderingTicks: function (args: SliderTickEventArgs) {
    let totalMiliSeconds = Number(args.value);
    // Convert the current milliseconds to the respective date in
    desired format
    let custom = { year: "numeric", month: "short", day: "numeric" };
    args.text = new Date(totalMiliSeconds).toLocaleDateString("en-us",
    custom);
  },
  ticks: {
    placement: 'After',
    // 2 * 86400000 milliseconds for two days
    largeStep: 2 * 86400000
  },
  showButtons: true
});
// Render initialized Slider
dateObj.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 Slider</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Slider Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="slider">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 260px;
}

```

```
margin: 0 auto;
padding: 30px 10px;
width: 460px;
}
```

Achieve time format

The time formatting can be achieved in the same manner as date formatting using `renderingTicks` and `change` events. The process of time formatting is explained in the following sample.

INDEX.TS

```
import { Slider, SliderTickEventArgs, SliderTooltipEventArgs } from
 '@syncfusion/ej2-inputs';
// Initialize slider control
let timeObj: Slider = new Slider({
  /**
   * Initialize the min and max values of the slider using JavaScript date
   functions
   * new Date (Year, Month, day, hours, minutes, seconds, milliseconds)
   */
  min: new Date(2013, 6, 13, 11).getTime(),
  max: new Date(2013, 6, 13, 17).getTime(),
  value: new Date(2013, 6, 13, 13).getTime(),
  // 3600000 milliseconds = 1 Hour
  step: 3600000,
  tooltipChange: function (args: SliderTooltipEventArgs) {
    let totalMiliSeconds = Number(args.text);
    let custom = { hour: '2-digit', minute: '2-digit' };
    args.text = new Date(totalMiliSeconds).toLocaleTimeString("en-us",
custom);
  },
  tooltip: {
    placement: 'Before',
    isVisible: true
  },
  renderingTicks: function (args: SliderTickEventArgs) {
    let totalMiliSeconds = Number(args.value);
    let custom = { hour: '2-digit', minute: '2-digit' };
    args.text = new Date(totalMiliSeconds).toLocaleTimeString("en-us",
custom);
  },
  ticks: {
    placement: 'After',
    // 2 * 3600000 milliseconds = 2 Hour
    largeStep: 2 * 3600000
  },
  showButtons: true
});
// Render initialized slider
timeObj.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Slider Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="slider">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    height: 260px;
    margin: 0 auto;

```

```
padding: 30px 10px;
width: 460px;
}
```

Customize numeric Value

The numeric values can be formatted into different decimal digits or fixed number of whole numbers or to represent units. The numeric processing is demonstrated as follows.

INDEX.TS

```
import { Slider } from '@syncfusion/ej2-inputs';
// Initialize slider control
let kilometerObj: Slider = new Slider({
    min: 0, max: 100, step: 1, value: 30,
    // Apply two decimal specifiers formatting if any decimal values are
    // processed with 'Km' text appended to the value
    tooltip: { isVisible: true, format: '##.## Km' },
    // Apply two decimal specifiers formatting if any decimal values are
    // processed with 'Km' text appended to the value
    ticks: { placement: 'After', format: '##.## Km', largeStep: 20,
    smallStep: 10, showSmallTicks: true }
});
// Render initialized slider
kilometerObj.appendTo('#slider');
// Initialize slider control
let decimalObj: Slider = new Slider({
    min: 0.1, max: .2, step: 0.01, value: 0.13,
    // Apply three decimal specifiers formatting if any decimal values are
    // processed then reset will be appened with two zero
    tooltip: { isVisible: true, format: '##.##00' },
    // Apply three decimal specifiers formatting if any decimal values are
    // processed then reset will be appened with two zero
    ticks: { placement: 'After', format: '##.##00', largeStep: 0.02,
    smallStep: 0.01, showSmallTicks: true }
});
// Render initialized slider
decimalObj.appendTo('#slider1');
// Initialize slider control
let sliderObj: Slider = new Slider({
    min: 0, max: 100, step: 1, value: 30,
    // Apply numeric formatting with two leading zero for tooltip
    tooltip: { isVisible: true, format: '00##' },
    // Apply numeric formatting with two leading zero for ticks
    ticks: { placement: 'After', format: '00##', largeStep: 20, smallStep:
10, showSmallTicks: true }
});
// Render initialized slider
sliderObj.appendTo('#slider2');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Slider</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Essential JS 2 Slider Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

        <div class="wrap">
            <div class="label">Slider formatted with unit
representation</div>
            <div id="slider">
                </div>
            </div>

            <div class="wrap">
                <div class="label">Slider formatted with three decimal
specifiers</div>
                <div id="slider1">
                    </div>
            </div>

            <div class="wrap">
                <div class="label">Slider formatted with two leading zeros</div>
                <div id="slider2">
                    </div>
            </div>
        </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}

```



```
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  height: 100px;
  margin: 0 auto;
  padding: 30px 10px;
  width: 460px;
}
.wrap .label {
  text-align: center;
}
```

Customize the bar in EJ2 JavaScript Range slider control

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the slider bar. The slider bar can be customized with different themes. By default, slider have class name `e-slider-track` for bar. The class can be overridden with our own color values like the following code snippet.

```
.e-control.e-slider .e-slider-track .e-range {
background: linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%,
#e33df9 83%, #e14423 100%);
}
```

You can also apply background color for a certain range depending upon slider values, using change event.

```
`ts
change: (args: SliderChangeEventArgs) => {
  if (args.value > 0 && args.value <= 25) {
    // Change handle and range bar color to green when
    (sliderHandle as HTMLElement).style.backgroundColor = 'green';
    (sliderTrack as HTMLElement).style.backgroundColor = 'green';
  } else if (args.value > 25 && args.value <= 50) {
    // Change handle and range bar color to royal blue
    (sliderHandle as HTMLElement).style.backgroundColor = 'royalblue';
```

```

(sliderTrack as HTMLElement).style.backgroundColor = 'royalblue';
} else if (args.value > 50 && args.value <= 75) {
// Change handle and range bar color to dark orange
(sliderHandle as HTMLElement).style.backgroundColor = 'darkorange';
(sliderTrack as HTMLElement).style.backgroundColor = 'darkorange';
} else if (args.value > 75 && args.value <= 100) {
// Change handle and range bar color to red
(sliderHandle as HTMLElement).style.backgroundColor = 'red';
(sliderTrack as HTMLElement).style.backgroundColor = 'red';
}
}
,

```

INDEX.TS

```

import { Slider, SliderChangeEventArgs } from '@syncfusion/ej2-inputs';
// Initialize Slider component
let heightSlider: Slider = new Slider({
// Set the value for slider
value: 30,
min: 0, max: 100
});
heightSlider.appendTo('#height_slider');
// Initialize Slider component
let gradientSlider: Slider = new Slider({
// Set slider minimum and maximum values
min: 0, max: 100,
// Set the initial range values for slider
value: 50,
type: 'MinRange'
});
gradientSlider.appendTo('#gradient_slider');
let sliderTrack: HTMLElement;
let sliderHandle: HTMLElement;
// Initialize Slider component
let dynamicColorSlider: Slider = new Slider({
// Set slider minimum and maximum values
min: 0, max: 100,
value: 20,
type: 'MinRange',
// Handler used for slider created event
created: () => {
sliderTrack = (document.getElementById('dynamic_color_slider')
as HTMLElement).querySelector('.e-range');
sliderHandle = (document.getElementById('dynamic_color_slider')
as HTMLElement).querySelector('.e-handle');
(sliderHandle as HTMLElement).style.backgroundColor = 'green';
(sliderTrack as HTMLElement).style.backgroundColor = 'green';
},

```

```

// Handler used for slider change event
change: (args: SliderChangeEventArgs) => {
    if (args.value > 0 && args.value <= 25) {
        // Change handle and range bar color to green when
        (sliderHandle as HTMLElement).style.backgroundColor =
'green';
        (sliderTrack as HTMLElement).style.backgroundColor =
'green';
    } else if (args.value > 25 && args.value <= 50) {
        // Change handle and range bar color to royal blue
        (sliderHandle as HTMLElement).style.backgroundColor =
'royalblue';
        (sliderTrack as HTMLElement).style.backgroundColor =
'royalblue';
    } else if (args.value > 50 && args.value <= 75) {
        // Change handle and range bar color to dark orange
        (sliderHandle as HTMLElement).style.backgroundColor =
'darkorange';
        (sliderTrack as HTMLElement).style.backgroundColor =
'darkorange';
    } else if (args.value > 75 && args.value <= 100) {
        // Change handle and range bar color to red
        (sliderHandle as HTMLElement).style.backgroundColor = 'red';
        (sliderTrack as HTMLElement).style.backgroundColor = 'red';
    }
}
});
dynamicColorSlider.appendTo('#dynamic_color_slider');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div class="col-lg-12 control-section">
            <div class="slider-content-wrapper">
                <div class="slider_container">
                    <div class="slider-labeltext
slider_userselect">Height</div>
                    <!-- Square slider element -->
                    <div id="height_slider"></div>
                </div>
                <div class="slider_container">
                    <div class="slider-labeltext slider_userselect">Gradient
color</div>
                    <div id="gradient_slider"></div>
                </div>
                <div class="slider_container">
                    <div class="slider-labeltext slider_userselect">Dynamic
thumb and selection bar color</div>
                    <div id="dynamic_color_slider"></div>
                </div>
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: visible;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.slider-content-wrapper {
    width: 40%;
    margin: 0 auto;
    min-width: 185px;
}
.slider-userselect {
    -webkit-user-select: none;
    /* Safari 3.1+ */
    -moz-user-select: none;
    /* Firefox 2+ */
    -ms-user-select: none;
}

```

```

/* IE 10+ */
user-select: none;
/* Standard syntax */
}
.slider-labeltext {
  text-align: left;
  font-weight: 500;
  font-size: 13px;
  padding-bottom: 10px;
}
#height_slider .e-handle, #gradient_slider .e-handle {
  height: 20px;
  width: 20px;
  margin-left: -10px;
  top: calc(50% - 10px);
}
.slider_container {
  margin-top: 40px;
}
#height_slider .e-tab-handle::after {
  background-color: #f9920b;
}
#height_slider .e-slider-track {
  height: 8px;
  top: calc(50% - 4px);
  border-radius: 0;
}
#gradient_slider .e-range {
  height: 6px;
  top: calc(50% - 3px);
  border-radius: 5px;
  background: -moz-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
  background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#1e5799), color-stop(100%,#7db9e8));
  background: -webkit-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
  background: linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
  background: -o-linear-gradient(left, #e1451d 0%,#e14423 100%);
  background: -ms-linear-gradient(left, #e1451d 0%,#e14423 100%);
}
#gradient_slider .e-slider-track {
  height: 8px;
  top: calc(50% - 4px);
  border-radius: 5px;
}

```

Customize the ticks in EJ2 JavaScript Range slider control

Slider view can be customized via CSS. By overriding the slider CSS classes, you can customize the ticks. The ticks in slider allows you to easily identify the current value/values of the slider. It contains [smallStep](#) and [largeStep](#). By default, slider has class `e-tick` for slider ticks. You can override the class as per your requirement. Refer to the following code snippet to render ticks.

```
`ts
.e-scale .e-tick.e-custom::before {
content: '\e967';
position: absolute;
}
`
```

Here, the color for rendered ticks has been applied through `nth-child(childnumber)`. *The color is applied to the value of the childnumber in the slider.*

```
`ts
ticks_slider .e-scale :nth-child(1)::before {
color: red;
}
`
```

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { Slider, SliderTickRenderedEventArgs, SliderTickEventArgs } from
'@syncfusion/ej2-inputs';
// Initialize slider component
let ticksSlider: Slider = new Slider({
// Set slider minimum and maximum values
min: 0, max: 100,
// Set the initial value for slider
value: 30,
// Set the step value for slider
step: 5,
// Set the type to render minRange slider
type: 'MinRange',
// Initialize ticks with placement, largeststep
ticks: { placement: 'Before', largestStep: 20 },
// Handler used to add custom class to all tick element
renderingTicks: (args: SliderTickEventArgs) => {
if (args.tickElement.classList.contains('e-large')) {
args.tickElement.classList.add('e-custom');
}
}
});
ticksSlider.appendTo('#ticks_slider');
// Initialize slider component
let customTicks: Slider = new Slider({
// Set slider minimum and maximum values
min: 0, max: 100,
// Set the initial value for slider
value: 30,
// Set the type to render minRange slider
type: 'MinRange',
// Initialize ticks with placement, largeststep, smallstep
```

```

        ticks: { placement: 'Both', largeStep: 20, smallStep: 5 },
        // Handler used to customize tick element
        renderedTicks: (args: SliderTickRenderedEventArgs) => {
            let li: any = args.ticksWrapper.getElementsByClassName('e-
large');
            let remarks: any = ['Very Poor', 'Poor', 'Average', 'Good',
'Very Good', 'Excellent'];
            for (let i: number = 0; i < li.length; ++i) {
                (li[i].querySelectorAll('.e-tick-both')[1] as
HTMLElement).innerText = remarks[i];
            }
        });
        customTicks.appendTo('#slider');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Slider</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Slider Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="col-lg-12 control-section">
            <div class="slider-content-wrapper">
                <div class="slider_container" id="slider_wrapper">
                    <div class="slider_labelText userselect">Dynamic ticks
color</div>

                    <!-- Ticks slider element -->
                    <div id="ticks_slider"></div>
                </div>
                <div class="slider_container">
                    <div class="slider_labelText userselect">Ticks with
legends</div>

                    <!-- Ticks slider element -->
                    <div id="slider"></div>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c9f;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.slider-content-wrapper {
    width: 40%;
    margin: 0 auto;
    min-width: 185px;
}
.userselect {
    -webkit-user-select: none;
    /* Safari 3.1+ */
    -moz-user-select: none;
    /* Firefox 2+ */
    -ms-user-select: none;
    /* IE 10+ */
    user-select: none;
    /* Standard syntax */
}
.slider_labelText {
    text-align: left;
    font-weight: 500;
    font-size: 13px;
    padding-bottom: 40px;
}
.slider_container {
    margin-top: 40px;
}
.e-bigger .slider-content-wrapper {
    width: 80%;
}
#ticks_slider .e-range {
    z-index: unset;
}

```



```

}
/* csslint ignore:start */
@font-face {
  font-family: 'e-customized-icons';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj8iS4cAAAEoAAAAVmNtYXDS5tJrAAABjAAAAEBnbHlmdMA
KbQAAAdQAAAOwaGVhZBNseyYAAADQAAANmhoZWEHogNjAAAArAAAACRobXR4C9AAAAAAYAAAAA
MbG9jYQCaAdgAAAHMAAACG1heHABEAeEuAAABCAAAACBuYW1lc0cOBgAABYQAAAIlcG9zdNSlKbQ
AAAEsAAAAARwABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAAwABAAAAQAAtxzLE18
PPPUACwPoAAAAANgtmycAAAAA2C2bJwAAAAAD8wPzAAAAACAACAAAAAAAAAAAAEAAAADASIaAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPwAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6QLpZwNS/2oAWgPzAJYAAAAABAAAAAAAAAAAAAPoAAA
D6AAAAAAAAAgAAAAAAUAAAAQAABQABAAsAAAAABgAEAAEAukC6Wf//wAA6QLpZ///AAAAAA
BAAYABgAAAAEAAgAAAAAAMgHYAAIAAAAAA+oD6gAzAIcAAAEzHxghNT8WEx8THQEPEisBLxI9AT8
SAGAQECQmKCgpKScTEhIREA8ODQwKcGqHBQBAfwqAQMFbGcKCgWNDg8QERISEycpKSgoJiQgDQw
MDAwXFhUUEhEPDQsJCAIDAQEBAQMCCakLDQ8REhQVfHcMDAwMDQ0MDAwMFxYVFBIRDw0LCQgCAwE
BAQEDAggJCw0PERIUFYXDAwMDAGFAQMEBwkKDQ4ICakKCgoLCwwMDAcNDg8Og3sPDw4NDgwMDAs
LCgoKCQgIDg0KCQcEawJnAQEBAGMHCgsNDxESExUWFwwMDQwNDA0MDAwXFhUTExAPDQwJBwMCAGe
BAGIDBwkMDQ8QExMVfHcMDAwNDA0MDQwMFxYVExIRDw0LCgcDAGEBAAAAwAAAAAD8wPzAF8AwAE
hAAABDxmFfZ8XLxcPAjcfFA8XLxc/Fx8CJw8UHxc/Fy8XDwIBqRQUFBISERAQDg0NCwoJBwCFBAI
BAQIEBQCcHCQoLDQ00EBAREhIUFBQVfHcYWFhYWFRUTFBISERAQDg0NCwoJBwCFBAIABAQIEBQCcHCQo
LDQ00EBAREhIUExUVfHcYWFhYWTg4NGxkZGBYWFrmSEA8OCwsIBwUDAQEDBQCICwsODxASExUWFhg
ZGRsbHB0dHh4dHRwbGxkZGBYWFrmSEA8NDAsIBwUDAQEDBQCICwsODxASExUVfXgZGRsbHB0dHh4
dHd0QDx4eHBsaGRcWFRIREA0MCQgGawEBawYICQwNEBESFRYXGRobHB4eHyEgIiIiIiAhHx4eHBs
aGRcWFRIREA0MCQgGawEBawYICQwNEBESFRYXGRobHB4eHyEgIiIiIiEDPAYICQoLDQ00EBAREhI
TFBUVFRYXfHcYWFhYWFRUTFBISERAQDg0MDA0JBwCFBAIABAQIEBQCcHCQoMDA00EBAREhIUFBQVfHcYWFxY
VFRUUEXISERAQDg0NCwoJCAYFBAIABAQIEZAQECgwODxASExUVfXgYGHsbHB0dHh4dHRwbGxkZGBY
WFBQSEA8NDAsIBwUDAQEDBQCICwsODxASExUWFhgZGRsbHB0dHh4dHRwbGxoYGBcVFRMSEA8OCws
IBwUDAQEDBTYFBQwNEBESFRYXGRobHB0fHyEgIiIiIiEgHx4eHBsaGRcWFBMRDw4MCQgGawEBawY
ICQwODxETFBYXGRobHB4eHyEgIiIiIiAhHx4eHBsaGRcWFRIRDw4MCQgGawEBawYAAAAAAAAASAN4
AAQAAAAAAAAABAAAAQAAAAAAAAAHAAEAQAAAAAAAAAQAHAAGAAQAAAAAAAAAAHAA8AAQAAAAAAAABAA
HABYAAQAAAAAAAABQALAB0AAQAAAAAAAAABgAHACgAAQAAAAAAAAACgAsAC8AAQAAAAAAAAACwASAFsAAwABBak
AAAACAG0AAwABBakAAQAOAG8AAwABBakAAgAOAH0AAwABBakAAwAOAISAAwABBakABAAOAJkAAwA
BBakABQAWAKcAAwABBakABgAOAL0AAwABBakACgBYAMsAAwABBakACwAkASMgZS1pY29uc1JlZ3V
sYXJlLW1lb25zZS1pY29uc1ZlcnNpb24gMS4wZS1pY29uc0ZvbG9ja2Z2VuZXJhdGVkIHVzaW5nIFN
5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwB
SAGUAZwB1AGwAYQByAGUALQBpAGMAbwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACA
AMQAuADAAZQAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQByAGEAdABLAGQAIAB1AHMAaQB
uAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHc
ALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAAAAAAKAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAADAQIBAwEEAAh0ZW1wLWN1cxJGQl9DaGVja2JveF9zZWx1Y3QAAAA=)
format('trueType');
  font-weight: normal;
  font-style: normal;
}
/* csslint ignore:end */
#ticks_slider .e-scale .e-tick {
  background-image: none;
  visibility: visible;
  font-family: 'e-customized-icons';
}
#ticks_slider .e-scale {
  z-index: 0;
}
#ticks_slider .e-scale .e-custom::before {
  content: '\e967';
  position: absolute;

```

```

}
#ticks_slider .e-scale :nth-child(1)::before {
  color: red;
}
#ticks_slider .e-scale :nth-child(2)::before {
  color: blue;
}
#ticks_slider .e-scale :nth-child(3)::before {
  color: green;
}
#ticks_slider .e-scale :nth-child(4)::before {
  color: blueviolet;
}
#ticks_slider .e-scale :nth-child(5)::before {
  color: orange;
}
#ticks_slider .e-scale :nth-child(6)::before {
  color: pink;
}
#ticks_slider .e-scale .e-custom::before {
  font-size: 10px;
}
#ticks_slider .e-scale .e-custom::before {
  top: calc(50% + 1px);
  left: calc(50% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(1)::before {
  top: calc(50% + 1px);
  left: calc(0% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(6)::before {
  top: calc(50% + 1px);
  left: calc(100% - 6px);
}

```

Customize the limits in EJ2 JavaScript Range slider control

Slider appearance can be customized via CSS. By overriding the slider CSS classes, the slider limit bar can be customized. Here, the limit bar is customized with different background color. By default, the slider has class `e-limits` for limits bar. You can override the class with our own color values as given in the following code snippet.

```

.e-slider-container.e-horizontal .e-limits {
background-color: rgba(69, 100, 233, 0.46);
}

```

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { Slider, NumericTextBox, ChangeEventArgs } from '@syncfusion/ej2-inputs';

```

```

import { CheckBox, ChangeEventArgs as CheckBoxChange } from
 '@syncfusion/ej2-buttons';
// tslint:disable-next-line:max-func-body-length
// Initialize slider control
let minrangeObj: Slider = new Slider({
    // Set slider minimum and maximum values
    min: 0, max: 100,
    // Set the value for slider
    value: 30,
    // Set the step value
    step: 1,
    // Initialize ticks with placement, largeststep, smallstep
    ticks: { placement: 'Before', largeStep: 20, smallStep: 5,
showSmallTicks: true },
    // Set the type for slider
    type: 'MinRange',
    // Set the limit values for the slider
    limits: { enabled: true, minStart: 10, minEnd: 40 },
    // Initialize tooltip with placement and showOn
    tooltip: { isVisible: true, placement: 'Before', showOn: 'Focus' }
});
minrangeObj.appendTo('#minrange');
// Initialize slider control
let rangeObj: Slider = new Slider({
    // Set slider minimum and maximum values
    min: 0, max: 100,
    // Set the initial range values for slider
    value: [30, 70],
    // Set the step value
    step: 1,
    // Set the type to render range slider
    type: 'Range',
    // Initialize ticks with placement, largeststep, smallstep
    ticks: { placement: 'Before', largeStep: 20, smallStep: 5,
showSmallTicks: true },
    // Set the limit values for the slider
    limits: { enabled: true, minStart: 10, minEnd: 40, maxStart: 60,
maxEnd: 90 },
    // Initialize tooltip with placement and showOn
    tooltip: { isVisible: true, placement: 'Before', showOn: 'Focus' }
});
rangeObj.appendTo('#range');
// Initialize NumericTextBox
let minStart: NumericTextBox = new NumericTextBox({
    value: 10,
    min: 0,
    max: 100,
    change: minStartChange
});
minStart.appendTo('#minStart');
let minEnd: NumericTextBox = new NumericTextBox({
    value: 40,
    min: 0,
    max: 100,
    change: minEndChange
});
minEnd.appendTo('#minEnd');

```

```

let maxStart: NumericTextBox = new NumericTextBox({
    value: 60,
    min: 0,
    max: 100,
    change: maxStartChange
});
maxStart.appendTo('#maxStart');
let maxEnd: NumericTextBox = new NumericTextBox({
    value: 90,
    min: 0,
    max: 100,
    change: maxEndChange
});
maxEnd.appendTo('#maxEnd');
// Initialize Checkbox
let fixedOne: CheckBox = new CheckBox({ change: fixOne });
fixedOne.appendTo('#fixedOne');
let fixedSecond: CheckBox = new CheckBox({ change: fixSecond });
fixedSecond.appendTo('#fixedSecond');
// Eventlisteners to lock first handle of the both sliders
function fixOne(args: CheckBoxChange): void {
    minrangeObj.limits.startHandleFixed = args.checked;
    rangeObj.limits.startHandleFixed = args.checked;
}
// Eventlisteners to lock second handle of the both sliders
function fixSecond(args: CheckBoxChange): void {
    minrangeObj.limits.endHandleFixed = args.checked;
    rangeObj.limits.endHandleFixed = args.checked;
}
// Eventlisteners to change limit values for both sliders
function minStartChange(args: ChangeEventArgs): void {
    minrangeObj.limits.minStart = args.value;
    rangeObj.limits.minStart = args.value;
}
function minEndChange(args: ChangeEventArgs): void {
    minrangeObj.limits.minEnd = args.value;
    rangeObj.limits.minEnd = args.value;
}
function maxStartChange(args: ChangeEventArgs): void {
    minrangeObj.limits.maxStart = args.value;
    rangeObj.limits.maxStart = args.value;
}
function maxEndChange(args: ChangeEventArgs): void {
    minrangeObj.limits.maxEnd = args.value;
    rangeObj.limits.maxEnd = args.value;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Slider</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Slider Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="col-lg-8 control-section">
            <div class="content-wrapper">
                <div class="sliderwrap">
                    <label class="userselect">MinRange Slider With
Limits</label>
                    <div id="minrange"></div>
                </div>
                <div class="sliderwrap">
                    <label class="userselect">Range Slider With
Limits</label>
                    <div id="range"></div>
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

```

}
.content-wrapper {
  width: 52%;
  margin: 0 auto;
  min-width: 185px;
}
.sliderwrap {
  margin-top: 45px;
}
.e-bigger .content-wrapper {
  width: 80%;
}
.sliderwrap label {
  padding-bottom: 50px;
  font-size: 13px;
  font-weight: 500;
  margin-top: 15px;
  display: block;
}
.userselect {
  -webkit-user-select: none;
  /* Safari 3.1+ */
  -moz-user-select: none;
  /* Firefox 2+ */
  -ms-user-select: none;
  /* IE 10+ */
  user-select: none;
  /* Standard syntax */
}
.property-custom td {
  padding: 5px;
}
#range .e-limits, #minrange .e-limits {
  background-color: #ccc;
  background-color: rgba(69, 100, 233, 0.46);
}

```

Customize the thumb in EJ2 JavaScript Range slider control

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the thumb. By default, slider has unique class `e-handle` for slider thumb. You can override the following class as per your requirement. Here, in the sample, the slider thumb has been customized to square, circle, oval shapes, and background image has also been customized.

```

`ts
.e-control.e-slider .e-handle {
  background-image: url('https://ej2.syncfusion.com/demos/src/slider/images/thumb.png');
  background-color: transparent;
  height: 25px;
  width: 25px;
}

```

```

square_slider.e-control.e-slider .e-handle {
border-radius: 0%;

background-color: #f9920b;

border: 0;

}

circle_slider.e-control.e-slider .e-handle {
border-radius: 50%;

background-color: #f9920b;

border: 0;

}

oval_slider.e-control.e-slider .e-handle {
height: 25px;

width: 8px;

top: 3px;

border-radius: 15px;

background-color: #f9920b;

}
,

```

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { Slider } from '@syncfusion/ej2-inputs';
    // Initialize slider control
    let squareSlider: Slider = new Slider({
        // Set the value for slider
        value: 30,
        min: 0, max: 100
    });
    squareSlider.appendTo('#square_slider');
    // Initialize slider control
    let circleSlider: Slider = new Slider({
        // Set the value for slider
        value: 30,
        // Set slider minimum and maximum values
        min: 0, max: 100
    });
    circleSlider.appendTo('#circle_slider');
    // Initialize slider control
    let ovalSlider: Slider = new Slider({
        // Set the value for slider
        value: 30,
        // Set slider minimum and maximum values
        min: 0, max: 100
    });

```

```

ovalSlider.appendTo('#oval_slider');
// Initialize slider control
let imageSlider: Slider = new Slider({
  // Set the value for slider
  value: 30,
  // Set slider minimum and maximum values
  min: 0, max: 100,
  ticks: { placement: 'After' }
});
imageSlider.appendTo('#image_slider');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="col-lg-12 control-section">
      <div class="slider-content-wrapper">
        <div class="slider_container">
          <div class="labelText slider-userselect">Square</div>
          <!-- Square slider element -->
          <div id="square_slider"></div>
        </div>
        <div class="slider_container">
          <div class="labelText slider-userselect">Circle</div>
          <!-- Circle slider element -->
          <div id="circle_slider"></div>
        </div>
        <div class="slider_container">
          <div class="labelText slider-userselect">Oval</div>
          <!-- Oval slider element -->
          <div id="oval_slider"></div>
        </div>
        <div class="slider_container">

```



```

        <div class="labelText slider-userselect">Custom
image</div>
        <!-- Image slider element -->
        <div id="image_slider"></div>
    </div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c00;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.slider-content-wrapper {
    width: 40%;
    margin: 0 auto;
    min-width: 185px;
}
.slider-userselect {
    -webkit-user-select: none;
    /* Safari 3.1+ */
    -moz-user-select: none;
    /* Firefox 2+ */
    -ms-user-select: none;
    /* IE 10+ */
    user-select: none;
    /* Standard syntax */
}
.labelText {
    text-align: left;
    font-weight: 500;
    font-size: 13px;
    padding-bottom: 10px;
}
.slider_container {
    margin-top: 40px;
}

```

```

.e-bigger .content-wrapper {
  width: 80%;
}
#square_slider .e-handle {
  border-radius: 0;
  background-color: #f9920b;
  border: 0;
}
#circle_slider .e-handle {
  background-color: #f9920b;
  border-radius: 50%;
  border: 0;
}
#image_slider .e-handle {
  height: 25px;
  width: 24px;
  background-size: 24px;
}
#image_slider .e-handle {
  background-image:
url('https://ej2.syncfusion.com/demos/src/slider/images/thumb.png');
  background-repeat: no-repeat;
  background-color: transparent;
  border: 0;
}
#square_slider .e-tab-handle::after,
#circle_slider .e-tab-handle::after {
  background-color: #f9920b;
}
#image_slider .e-tab-handle::after {
  background-color: transparent;
}
#oval_slider .e-handle {
  height: 25px;
  width: 8px;
  top: 3px;
  border-radius: 15px;
  background-color: #f9920b;
}

```

Validate slider using formvalidator in EJ2 JavaScript Range slider control

The Slider component can be validated using our [FormValidator](#). The following steps walk-through slider validation.

- Render slider component inside a form.
- Bind [changed](#) event in the slider component to validate the slider value when the value changes.
- Initialize and render FormValidator for the form using form ID.

`ts

```
// Initialize Form validation
```

```
let formObj: FormValidator;
```

```
formObj = new FormValidator("#formId", options);
```

- Set the required property in the FormValidator [rules](#) collection. Here, the [min](#) property of slider that sets the minimum value in the slider component is set, and it has hidden input as enable `validateHidden` property is set to true.

```
`ts
// Slider element
<div id="default" name="slider"></div>
// sets required property in the FormValidator rules collection
let options: FormValidatorModel = {
  rules: {
    'default': {
      validateHidden: true,
      min: [6, "You must select value greater than 5"]
    }
  }
};
```

Form validation is done either by ID or name value of the slider component. Above ID of the slider is used to validate it.

Using slider name: Render slider with name attribute. In the following code snippet, name attribute value of slider is used for form validation.

```
`ts
// Slider element
<div id="default" name="slider"></div>
// sets required property in the FormValidator rules collection
let options: FormValidatorModel = {
  rules: {
    'slider': {
      validateHidden: true,
      min: [6, "You must select value greater than 5"]
    }
  }
};
```

- Validate the form using [validate](#) method, and it validates the slider value with the defined rules collection and returns the result. If user selects the value less than the minimum value, form will not submit.

```
`ts
```

```
formObj.validate();
```

- Slider validation can be done during value changes in slider. Refer to the following code snippet.

```
`ts
```

```
// change event handler for slider
```

```
function onChanged(args: any) {
```

```
formObj.validate();
```

```
}
```

The **FormValidator** has following default validation rules, which are used to validate the Slider component.

Rules	Description	Example
max	Slider component must have value less than or equal to max number	if max: 3, 3 is valid and 4 is invalid
min	Slider component must have value greater than or equal to min number	if min: 4, 5 is valid and 2 is invalid
regex	Slider component must have valid value in regex format	if regex: '/4/', 4 is valid and 1 is invalid
range	Slider component must have value between range number	if range: [4,5], 4 is valid and 6 is invalid

INDEX.TS

```
import { Slider } from '@syncfusion/ej2-inputs';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
// Initialize Slider component
let SliderMinObj: Slider = new Slider({
  type: 'MinRange',
  value: 30,
  ticks: { placement: 'Before', largeStep: 20, smallStep: 5, showSmallTicks:
true },
  changed: onMinChanged
});
```

```
SliderMinObj.appendTo("#min-slider");
// sets required property in the FormValidator rules collection
let minOptions: FormValidatorModel = {
  rules: {
    'min-slider': {
      validateHidden: true,
      min: [40, "You must select value greater than or equal to 40"]
    }
  }
};
// Initialize Form validation
let formMinObj: FormValidator;
formMinObj = new FormValidator("#formMinId", minOptions);
function onMinChanged(args: any) {
  // validate the slider value in the form
  formMinObj.validate();
}
// Initialize Slider component
let SliderMaxObj: Slider = new Slider({
  type: 'MinRange',
  value: 30,
  ticks: { placement: 'Before', largeStep: 20, smallStep: 5, showSmallTicks:
true },
  changed: onMaxChanged
});
SliderMaxObj.appendTo("#max-slider");
// sets required property in the FormValidator rules collection
let maxOptions: FormValidatorModel = {
  rules: {
    'max-slider': {
      validateHidden: true,
      max: [40, "You must select value less than or equal to 40"]
    }
  }
};
// Initialize Form validation
let formMaxObj: FormValidator;
formMaxObj = new FormValidator("#formMaxId", maxOptions);
function onMaxChanged(args: any) {
  // validate the slider value in the form
  formMaxObj.validate();
}
// Initialize Slider component
let SliderValObj: Slider = new Slider({
  type: 'MinRange',
  value: 30,
  ticks: { placement: 'Before', largeStep: 20, smallStep: 5, showSmallTicks:
true },
  changed: onValChanged
});
SliderValObj.appendTo("#val-slider");
// sets required property in the FormValidator rules collection
let valOptions: FormValidatorModel = {
  rules: {
    'val-slider': {
      validateHidden: true,
      regex: [/40/, "You must select value equal to 40"]
    }
  }
};
```

```

    }
  }
};
// Initialize Form validation
let formValObj: FormValidator;
formValObj = new FormValidator("#formValId", valOptions);
function onValChanged(args: any) {
  // validate the slider value in the form
  formValObj.validate();
}
// Initialize Slider component
let SliderRangeObj: Slider = new Slider({
  type: 'MinRange',
  value: 30,
  ticks: { placement: 'Before', largeStep: 20, smallStep: 5, showSmallTicks:
true },
  changed: onRangeChanged
});
SliderRangeObj.appendTo("#range-slider");
// sets required property in the FormValidator rules collection
let rangeOptions: FormValidatorModel = {
  rules: {
    'range-slider': {
      validateHidden: true,
      range: [40, 80, "You must select values between 40 and 80"]
    }
  }
};
// Initialize Form validation
let formRangeObj: FormValidator;
formRangeObj = new FormValidator("#formRangeId", rangeOptions);
function onRangeChanged(args: any) {
  // validate the slider value in the form
  formRangeObj.validate();
}
// Initialize Slider component
let SliderCustomObj: Slider = new Slider({
  type: 'Range',
  value: [30, 70],
  ticks: { placement: 'Before', largeStep: 20, smallStep: 5, showSmallTicks:
true },
  changed: onCustomChanged
});
SliderCustomObj.appendTo("#custom-slider");
// sets required property in the FormValidator rules collection
let customOptions: FormValidatorModel = {
  rules: {
    'custom-slider': {
      validateHidden: true,
      range: [validateRange, "You must select values between 40 and 80"]
    }
  }
};
// Initialize Form validation
let formCustomObj: FormValidator;
formCustomObj = new FormValidator("#formCustomId", customOptions);
function onCustomChanged(args: any) {

```

```
// validate the slider value in the form
formCustomObj.validate();
}
function validateRange(args: any) {
    return (SliderCustomObj.value as number[])[0] >= 40 &&
    (SliderCustomObj.value as number[])[1] <= 80;
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Slider</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Slider Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="col-lg-12 control-section">
            <div class="content-wrapper" style="margin-bottom:
25px;overflow-x: hidden">
                <div class="form-title">
                    <span>Min</span>
                </div>
                <form id="formMinId" class="form-horizontal">
                    <div class="form-group">
                        <div class="e-float-input">
                            <div id="min-slider" name="min-slider"></div>
                        </div>
                    </div>
                </form>
                <div class="form-title">
                    <span>Max</span>
                </div>
                <form id="formMaxId" class="form-horizontal">
                    <div class="form-group">
                        <div class="e-float-input">
                            <div id="max-slider" name="max-slider"></div>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
```

```

        </div>
    </form>
    <div class="form-title">
        <span>Value</span>
    </div>
    <form id="formValId" class="form-horizontal">
        <div class="form-group">
            <div class="e-float-input">
                <div id="val-slider" name="val-slider"></div>
            </div>
        </div>
    </form>
    <div class="form-title">
        <span>Range</span>
    </div>
    <form id="formRangeId" class="form-horizontal">
        <div class="form-group">
            <div class="e-float-input">
                <div id="range-slider" name="range-
slider"></div>
            </div>
        </div>
    </form>
    <div class="form-title">
        <span>Custom</span>
    </div>
    <form id="formCustomId" class="form-horizontal">
        <div class="form-group">
            <div class="e-float-input">
                <div id="custom-slider" name="custom-
slider"></div>
            </div>
        </div>
    </form>
</div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c00;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
}

```



```

    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.highcontrast form {
    background: #000000
}
/* csslint ignore:start */
.highcontrast label.e-custom-label,
.highcontrast label.e-float-text,
.highcontrast label.salary,
.highcontrast input::placeholder,
.highcontrast .e-float-input label.e-float-text {
    color: #fff !important;
    line-height: 2.3;
}
/* csslint ignore:end */
.e-error,
.e-float-text {
    font-weight: 500;
}
table,
td,
th {
    padding: 5px;
}
.form-horizontal .form-group {
    margin-left: 0;
    margin-right: 0;
}
form {
    border: 1px solid #ccc;
    box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
    border-radius: 5px;
    background: #f9f9f9;
    padding: 23px;
    padding-bottom: 20px;
    margin: auto;
    max-width: 650px;
}
.form-title {
    width: 100%;
    text-align: center;
    padding: 10px;
    font-size: 16px;
    font-weight: 600;
    color: black;
}

```

Show slider from hidden state in EJ2 JavaScript Range slider control

This section demonstrates how-to render the Slider component in hidden state and make it visible in button click. We can initialize Slider in hidden state by setting the display as none.

In the sample, by clicking on the button, we can make the Slider visible from hidden state, and we must also call the [refresh](#) method of the Slider to render it properly based on its original dimensions.

INDEX.TS

```
import { Slider, SliderChangeEventArgs } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
// Initialize the Button component.
let button: Button = new Button({ content: 'Button' });
// Render initialized button.
button.appendTo('#element');
// Initialize Slider Component
let defaultObj: Slider = new Slider({
    // Set slider minimum and maximum values
    // new Date(Year, Month, day, hours, minutes, seconds, milliseconds)
    min: new Date(2013, 6, 13, 11).getTime(), max: new Date(2013, 6, 13,
17).getTime(),
    // 3600000 milliseconds = 1 Hour
    step: 3600000,
    //Set buttons for slider
    showButtons: true,
    // Set the initial range values for slider
    value: new Date(2013, 6, 13, 13).getTime(),
    // Bind Tooltip change event for custom formatting
    tooltipChange: tooltipChangeHandler,
    // Initialize tooltip with placement
    tooltip: {
        placement: 'Before', isVisible: true, cssClass: 'e-tooltip-
customization'
    },
    // Bind ticks event for custom formatting
    renderingTicks: renderingTicksHandler,
    // Initialize ticks with placement, largeststep, smallstep
    ticks: {
        placement: 'After',
        // 2 * 3600000 milliseconds = 2 Hour
        largeStep: 2 * 3600000,
        smallStep: 3600000, showSmallTicks: true
    },
    // Set the type to render range slider
    type: 'MinRange'
});
defaultObj.appendTo('#slider');
function tooltipChangeHandler(args: SliderTooltipEventArgs): void {
    /**
     * toLocaleTimeString is predefined javascript date function, which is
    used to
     * customize the date in different format
     */
    let custom: { [key: string]: string } = { hour: '2-digit', minute: '2-
digit' };
    // Splitting the range values from the tooltip using space into an
    array.
    if (args.text.indexOf('-') !== -1) {
        let totalMilliSeconds: string[] = args.text.split(' ');
        // First part is the first handle value
        let firstPart: string = totalMilliSeconds[0];
```

```

        // Second part is the second handle value
        let secondPart: string = totalMilliseconds[2];
        firstPart = new Date(Number(firstPart)).toLocaleTimeString('en-us',
custom);
        secondPart = new Date(Number(secondPart)).toLocaleTimeString('en-
us', custom);
    } else {
        args.text = new Date(Number(args.text)).toLocaleTimeString('en-us',
custom);
    }
}
function renderingTicksHandler(args: SliderTickEventArgs): void {
    let totalMilliseconds: number = Number(args.value);
    /**
     * toLocaleTimeString is predefined javascript date function, which is
used to
     * customize the date in different format
     */
    let custom: { [key: string]: string } = { hour: '2-digit', minute: '2-
digit' };
    // Assigning our custom text to the tick value.
    args.text = new Date(totalMilliseconds).toLocaleTimeString('en-us',
custom);
}
//Visible slider by clicking the button
document.getElementById('element').onclick = function () {
    let slider = document.getElementById("case");
    let ticks = document.getElementById("slider");
    slider.style.display = "block";
    ticks.ej2_instances[0].refresh();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Slider</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Slider Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>
  <!--button to show slider-->
  <button id="element" style="float:left;">Button</button>

  <div id="container">
    <div class="wrap">
      <div id="case">
        <div id="slider">
        </div>
      </div>
    </div>
  </div>
</div>
<style>
/* render slider in hidden state */
  #case{
    display: none;
  }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  height: 260px;
  margin: 0 auto;
  padding: 80px 10px;
  width: 460px;
}

```

Reversible Range Slider in EJ2 JavaScript

You can create a Range Slider rendered with values in reverse order by setting the [min](#) property to the maximum value and the [max](#) property to the minimum value. An example of how to achieve a reversible Range Slider is shown below

INDEX.TS

```
import { Slider } from '@syncfusion/ej2-inputs';
// Initialize Slider control
let sliderObj: Slider = new Slider({
  ticks: { placement: 'Before', largeStep: 20, smallStep: 5,
showSmallTicks: true },
  tooltip: { placement: 'Before', isVisible: true, showOn: 'Always' },
  type: 'Range',
  // vertical orientation
  orientation: 'Vertical',
  // Set maximum value to min
  min: 100,
  // Set minimum value to max
  max: 0,
  // Slider current value
  value: [30, 70]
});
// Render initialized Slider
sliderObj.appendTo('#slider');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Slider</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Slider Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <div id="slider">
      </div>
    </div>
  </div>
</script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

INDEX.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  height: 260px;
  margin: 0 auto;
  padding: 30px 10px;
  width: 260px;
}

```

Reversible order can be achieved with [Horizontal](#) orientation Range Slider by setting [enableRtl](#) as true.

Rating

Precision modes in EJ2 JavaScript Rating control

You can use the [precision](#) property of the rating control to provide ratings with varying levels of precision.

The precision types of Rating are as follows:

- Full: The rating is increased in whole number increments. For example, if the current rating is 2, the next possible ratings are 3, 4, and so on.
- Half: The rating is increased in increments of 0.5 (half). For example, if the current rating is 2.5, the next possible ratings are 3, 3.5, 4, and so on.
- Quarter: The rating is increased in increments of 0.25 (quarter). For example, if the current rating is 3.75, the next possible ratings are 4, 4.25, 4.5, and so on.
- Exact: The rating is increased in increments of 0.1. For example, if the current rating is 3.9, the next possible ratings are 4, 4.1, 4.2, and so on.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Full precision Rating control.
var rating1 = new ej.inputs.Rating({ value: 3, precision: 'Full' });

```

```
// Initialize the Half precision Rating control.
var rating2 = new ej.inputs.Rating({ value: 2.5, precision: 'Half' });
// Initialize the Quarter precision Rating control.
var rating3 = new ej.inputs.Rating({ value: 3.75, precision: 'Quarter' });
// Initialize the Exact precision Rating control.
var rating4 = new ej.inputs.Rating({ value: 2.3, precision: 'Exact' });
// Render initialized Rating.
rating1.appendTo('#rating1');
rating2.appendTo('#rating2');
rating3.appendTo('#rating3');
rating4.appendTo('#rating4');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <label>Full Precision</label><br>
      <input id="rating1"><br>
      <label>Half Precision</label><br>
      <input id="rating2"><br>
      <label>Quarter Precision</label><br>
      <input id="rating3"><br>
      <label>Exact Precision</label><br>
      <input id="rating4"><br>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Labels in EJ2 JavaScript Rating control

You can use the [showLabel](#) property to display a label that shows the current value of the rating. When the `showLabel` property is set to `true`, a label will be displayed.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ showLabel: true, value:3 });
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>
  <script type="text/x-jsrender" id="labelTemplate">
    <span>${value} out of 5</span>
  </script>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

</script>
```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
```

Label position

The rating control allows you to place the label on the top, bottom, left, or right side of the rating using the [labelPosition](#) property.

The following label positions are supported:

- Top: The label is placed on the top of the rating.
- Bottom: The label is placed on the bottom of the rating.
- Left: The label is placed on the left side of the rating.
- Right: The label is placed on the right side of the rating.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Label Position Left Rating control.
var rating1 = new ej.inputs.Rating({ value:3, showLabel:true,
labelPosition:"Left"});
// Initialize the Label Position Right Rating control.
var rating2 = new ej.inputs.Rating({ value:3, showLabel:true, });
// Initialize the Label Position Top Rating control.
var rating3 = new ej.inputs.Rating({ value:3, showLabel:true,
labelPosition:"Top" });
// Initialize the Label Position Bottom Rating control.
var rating4 = new ej.inputs.Rating({ value:3, showLabel:true,
labelPosition:"Bottom" });
// Render initialized Rating.
rating1.appendTo('#rating1');
rating2.appendTo('#rating2');
rating3.appendTo('#rating3');
rating4.appendTo('#rating4');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <label>Left Label Position</label><br>
      <input id="rating1"><br>
      <label>Right Label Position</label><br>
      <input id="rating2"><br>
      <label>Top Label Position </label><br>
      <input id="rating3"><br>
      <label>Bottom Label Position</label><br>
      <input id="rating4"><br>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
}

```

```

height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 50px auto;
text-align: center;
}

```

Label template

You can use the [labelTemplate](#) tag directive to specify a custom template for the **Label** of the rating. The current value of the rating will be passed as the **value** property in the template context when building the content of the label. This allows you to include dynamic information about the rating in the template.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ showLabel: true, labelTemplate:
'#labelTemplate', value:3 });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">

```

```

    </div>
  </div>
  <script type="text/x-jsrender" id="labelTemplate">
    <span>${value} out of 5</span>
  </script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 50px auto;
  text-align: center;
}

```

Tooltip in EJ2 JavaScript Rating control

The rating control supports tooltip to show additional information in rating items by setting the [showTooltip](#) property. If enabled, the tooltip appears when the user hovers over a rating item.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Tooltip Rating control.
var rating = new ej.inputs.Rating({ showTooltip:true, value:3 });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
    <script id="tooltipTemplate" type="text/x-jsrender">
        ${if(value==1)}
        <b>Angry</b> ${else if(value==2)}
        <b>Sad</b> ${else if(value==3)}
        <b>Neutral</b> ${else if(value==4)}
        <b>Good</b> ${else}
        <b>Happy</b>
        ${/if}
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {

```

```

margin: 50px auto;
text-align: center;
}
/* To change the radius of the tooltip corners. */
.customtooltip.e-tooltip-wrap {
border-radius: 3px;
}
/* To change the size of the tooltip content. */
.customtooltip.e-tooltip-wrap .e-tip-content {
font-size: 14px;
}
/* To change the border color and width for tooltip. */
.customtooltip.e-tooltip-wrap.e-popup {
border: 2px solid #000000;
}
/* To change the color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
border: 12px #9693
}
/* To change the top border color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-top: 9.5px solid #000000;
}

```

Tooltip template

You can use the [tooltipTemplate](#) tag directive to specify a custom template for the tooltip of the rating. The current value of the rating will be passed as the **value** property in the template context when building the content of the tooltip. This allows you to include dynamic information about the rating in the template.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Show Tooltip Rating control.
var rating = new ej.inputs.Rating({ showTooltip:true, value:3,
tooltipTemplate: '#tooltipTemplate' });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
    <script id="tooltipTemplate" type="text/x-jsrender">
        ${if(value==1)}
        <b>Angry</b> ${else if(value==2)}
        <b>Sad</b> ${else if(value==3)}
        <b>Neutral</b> ${else if(value==4)}
        <b>Good</b> ${else}
        <b>Happy</b>
        ${/if}
    </script>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
/* To change the radius of the tooltip corners. */
.customtooltip.e-tooltip-wrap {
    border-radius: 3px;
}

```

```

}
/* To change the size of the tooltip content. */
.customtooltip.e-tooltip-wrap .e-tip-content {
  font-size:14px;
}
/* To change the border color and width for tooltip. */
.customtooltip.e-tooltip-wrap.e-popup {
  border: 2px solid #000000;
}
/* To change the color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
  border: 12px #9693
}
/* To change the top border color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
  border-top: 9.5px solid #000000;
}

```

Tooltip customization

You can customize the appearance of the tooltips using the `cssClass` property of the rating control and by defining the custom styles for tooltip elements like the below example.

You can find more information about customizing the appearance of the tooltip in the [Tooltip Customization](#) documentation.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Custom Tooltip Rating control.
let rating = new ej.inputs.Rating({ cssClass: "customtooltip",
showTooltip:true, value:3 });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
    <script id="tooltipTemplate" type="text/x-jsrender">
        ${if(value==1)}
        <b>Angry</b> ${else if(value==2)}
        <b>Sad</b> ${else if(value==3)}
        <b>Neutral</b> ${else if(value==4)}
        <b>Good</b> ${else}
        <b>Happy</b>
        ${/if}
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
/* To change the radius of the tooltip corners. */
.customtooltip.e-tooltip-wrap {
    border-radius: 3px;
}
/* To change the size of the tooltip content. */
.customtooltip.e-tooltip-wrap .e-tip-content {
    font-size: 14px;
}
/* To change the border color and width for tooltip. */

```

```
.customtooltip.e-tooltip-wrap.e-popup {
  border: 2px solid #000000;
}
/* To change the color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
  border: 12px #9693
}
/* To change the top border color for arrow of the tooltip. */
.customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
  border-top: 9.5px solid #000000;
}
```

Selection in EJ2 JavaScript Rating control

The rating control allows users to rate something using a visual scale, and the selection state can be changed by the user clicking or tapping on the stars in the rating scale or through code. The rating control has a minimum value and a reset button, and provides customization options for the selected rating value and selection behavior.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3 });
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
```

```

        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c8f;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}

```

Min value

You can use the [min](#) property of the rating control to set the minimum possible rating value the user can select. If you set the `min` property to 2, then you will not be able to select a rating lower than 2.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ min:2 });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Rating</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript Rating Control">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}

```

Single selection

You can use the [enableSingleSelection](#) property of the rating control to select only one item at a time. When the `enableSingleSelection` property is set to `true`, only the selected item will be considered to be in the selected state, while all other items will be unselected.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3, enableSingleSelection:true });
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008c9f;
  font-family: 'Helvetica Neue', 'calibri';
  font-size: 14px;
  height: 40px;
```

```

left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 50px auto;
text-align: center;
}

```

Show or hide reset button

You can reset the rating value to its default by using the [allowReset](#) property of the rating control. When the `allowReset` property is set to `true`, a reset button will be shown that allows the user to reset the rating value to its default.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3, allowReset:true });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}

```

Templates in EJ2 JavaScript Rating control

The rating control allows you to customize the appearance of the rating items using templates. You can use templates to specify a custom layout for the rating items, which can include any content you want. This allows you to create a more customized and interactive rating experience for the user.

The rating control supports below templates for item customization.

- [emptyTemplate](#)
- [fullTemplate](#)

Empty (unrated) symbol template

To customize the appearance of **unrated** items, you can use the **emptyTemplate** tag directive. It allows you to specify the desired custom content for the unrated items. The **value** and **index** are available in the template context for accessing information about the un-rated item.

If the **fullTemplate** is not defined, the **emptyTemplate** will be used as the default for both rated and unrated items. You can apply custom styles to differentiate between the rated and unrated states of the items.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ emptyTemplate: '#emptyTemplate', value:3
});

```

```
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>
  <script type="text/x-jsrender" id="emptyTemplate">
    <span class='custom-font sf-rating-heart'></span>
  </script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
```



```

left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 50px auto;
text-align: center;
}
.e-rating-container .custom-font {
/* To add the background color for the font icon. */
background: linear-gradient(to right, rgb(254,87,133,255) var(--rating-
value), transparent var(--rating-value));
/* To clip the background to the icon (text) alone. */
background-clip: text;
-webkit-background-clip: text;
/* To make the background color visible instead of font color. */
-webkit-text-fill-color: transparent;
/* To provide a border for font icon. */
-webkit-text-stroke: 1px rgb(254,87,133,255);
}
@font-face {
font-family: 'rating';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjluSfQAAAEoAAAAVmNtYXNlEudaAAABjAAAADhnbHlm4Li
FsgAAAcwAAAJsAGVhZCKCSVKAADQAAAAANmhoZWEIUQQEAAAArAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQCMATYAAAHEAAAAACG1heHABDwCZAAABCAAAACBuYW1l75Kp8wAABDgAAAIzCg9zdDjyU90
AAAZUAAAAANwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAA2T6Kh18
PPPUACwQAAAAAAN+4AkeAAAAA37gCQAAAAAAD9APaAAAAACAACAAAAAAAAAAAAEAAAADAI0AAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wHnAgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAA
EAAAAAAAAAgAAAAAUAUAAAAQAAABQABAakAAAAABAEAAEOcC//8AAOcB//8AAAAABAAQAAAA
CAAEAAAAAAAAIwBNgABAAAAAPzA9oAfAAAEw8WFR8PPw41Lx4jDwwvDw8GqAwMDAsKCgoJCAgIBwY
GBQUEBAMCAgEBAQECAwMEBQULFSMhOVJliOxTOSEdFg0IBQQDAwIBAQEBAgIDBAQFQBQYGBwgICak
KCgoLDAwMDAwMDQwMDQwZGBgYFxFUVFBIRCAgGBwkLCwwNDg4PEBAQEREREhEODg4ODg4NA8IGBwc
ICakJCgoKCwsMCwwNDA0MDQ0ODQ0ODQ0ODQ0NDRUIMCtEX26P/V5FKycjFhQNDQ0ODQ0ODQ0NDgw
NDQwNCwwMCwoLCgoJCAkIBwcGBQUEAwMCAQECEBQYJCw4PERMKCgsMEQ8PDQ0LCwoICAYFBAMCAQE
BAgIEBAUAAGAAAAAD9APFAAMajAAANzMRIwEPaXUXDwwRMzcfBDcXPwo9AS8FPwsvCDc1Pwg1LwU
1Pw01LwkHJT8ENS8LIw8BDK2tAfKCCgQBAQEGBGGERERITIGkJKBAGIQc1Bx45k9sOBQgLDQsJBQM
EAgIECQYCAQEBAw4ECQgGBwMDAQEBAQMDAwkCAQEDFgsFBAQDAwICAgQECgEBAQQKBwcGBQUEAwM
BAQEBAUHCQUFBQYR/q0PCQQDAgEBAwMKDBUDBwYMCw0HB1oBhwHeAQUDA3YfCgQsOh0bHBovCQg
bDP6KAQEfAwEBAQIBAQMGCGoMBggICAUICQgLBQQEBAUDBgMHCAGMCACIBwYGBgUFCQCCBgIEDAk
GBQYHCQkKCQgIBwsEAgUDAgQEBAUFBgcHCAcGBgYGCgkIBgICAQEBAUYxGRobDQ0MDQsiHjEEBAI
EAQECAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEABgABAAEAAAAAAAAIABwAHAAEAAAAAAAAAMABgA
OAAEAAAAAAAAQABgAUAAEAAAAAAAAUACwAAAEAAAAAAAAAYABgAlAAEAAAAAAAAoALAArAAEAAAAAAAA
AEgBXAAMAAQQAIAAAAGBpAAMAAQQAIAEADABrAAMAAQQAIAIADgB3AAMAAQQAIAAMADACFAAMAAQQA
JAAQADACRAAMAAQQAIAUAUFgCdAAMAAQQAIAAYADACzAAMAAQQAIAAoAWAC/AAMAAQQAIAAsAJAEXIHJ
hdGluZ1JlZ3VsYXJyYXRpbmdyYXRpbmdWZXJzaW9uIDEuMHJhdGluZ0ZvbnQgZ2VuZXJhdGVkIHV
zaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAACgBhAHQAaQB
uAGCAUgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCA
AMQAuADAACgBhAHQAaQB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCAgB1AGCA
nACAAUwB5AG4AYwBmAHUAACwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4
AcwB5AG4AYwBmAHUAACwBpAG8AbgAuAGMAbwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAwEACQMBBAAFaGVhcnQFdGhlbWIAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}

```

```
[class^="sf-rating-"], [class*=" sf-rating-"] {
  font-family: 'rating' !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.sf-rating-heart:before {
  content: "\e702";
}
```

The current value of the rating item available in the template context as `value` and in the rating item element as CSS Variable(`--rating-value`) can be used to support precision in templates.

Full (rated) symbol template

To customize the appearance of **rated** items in the rating control, you can use the `fullTemplate` tag directive. This directive allows you to specify a custom layout for the rated items, which can include any content you desire. The `value` and `index` are available in the template context for accessing information about the rated item.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ fullTemplate: '#fullTemplate',
emptyTemplate: '#emptyTemplate', value: 3 });
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
    <script type="text/x-jsrender" id="emptyTemplate">
        <span class='custom-font sf-icon-empty-star'></span>
    </script>

    <script type="text/x-jsrender" id="fullTemplate">
        <span class='custom-font sf-icon-fill-star'></span>
    </script>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}

#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}

.wrap {
  margin: 50px auto;
  text-align: center;
}

.e-rating-container .custom-font {
  /* To change the icon font color. */
  color: rgb(255, 215, 0);
}

@font-face {
  font-family: 'rating-template';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfMAAAEoAAAVmNtYXNlEODaAAABjAAAADhnbHlm+ic
DjQAAAcwAAAE0aGVhZCK49ucAAADQAAAAANmhoZWEIUQQEAAAArAAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQAcAJoAAAHEAAAAACGlheHABDwBkAAABCAAAACBuYWI1lmYExxgAAAwAAAAKFcG9zdCH169Q
AAAWIAAAAAOAAABAAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAwABAAAAAOAAgPX4jF8
```

```

PPPUACwQAAAAAAN/TWPsAAAAA39NY+wAAAAAD9AP0AAAACAACAAAAAEEAAAADAFgAAgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAAUAUAMAAQAAABQABAakAAAABAAEAAEOcB//8AAOcA//8AAAABAAQAAAA
BAAIAAAAAABwAmgABAAAAAAP0A/QACQAAAQUTAYUFAXMlAwFn/qX6OwE1ATU7+v6lmQKsNf7//pa
srAFqAQE1AUgAAAIAAAAAAAA/QD5AAAdAFcAAAEfBAUPAxUTLwEjDwETNS8DJT8EJwMFDwQVHwIDBx8
EMzclBRczPwU1Az8CNS8DJQMvBisBDwUCYAICBgMHASCwBAMCGuoHCAjPpGgIDBLEBHgcGBgJiHXb
+uQgHBgQBAGTUHgECBAUHCakIAQ4BDgcJBAQEbwQDhdMEAgMFBgf+thYDagMEBAQEBAQEBAQEAWI
CnwMDBgIDNbAGBgYE/uCBAGKBAR0HBgYGsDQCBAYD3Fr+9jwDBAcHBQgIB9T+twUIBwcEawKVlQI
BAGIFBwgJAUnUBwgJCACFBD0BCgQDBAICAgEBAGICBAMAAAAAEgDeAAEAAAAAAAAAQAQAAAAEAAA
AAAEADwABAAEAAAAAAAAIABwAQAAEAAAAAAAAAMADwAXAAEAAAAAAAAAQADwAmAAEAAAAAAAAUACwAlAAE
AAAAAAAAYADwBAAEAAAAAAAAoALABPAEEAAAAAAsAEgB7AAMAAQQJAaaaaAgCNAAMAAQQJAEEAHgC
PAAMAAQQJAAlADgCtAAMAAQQJAAMAHgC7AAMAAQQJAQAHAHgDZAAMAAQQJAUAUAFgD3AAMAAQQJAAY
AHgENAAMAAQQJAaoAWAErAAMAAQQJAAsAJAGDIHJhdGluZy10ZW1wbGF0ZVJlZ3VsYXJyYXRpbmc
tdGVtcGxhdGVyYXRpbmctdGVtcGxhdGVWZXJzaW9uIDEuMHJhdGluZy10ZW1wbGF0ZUZvbnQgZ2V
uZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACA
AcgBhAHQAaQBwAGcALQB0AGUAbQBwAGwAYQB0AGUAUgBlAGcAdQBsaGEAcgByAGEAdABpAG4AZwA
tAHQAZQBtAHAAbABhAHQAZQBzAGEAdABpAG4AZwAtAHQAZQBtAHAAbABhAHQAZQBWAGUAcgBzAGk
AbwBuACAAMQAuADAACgBhAHQAaQBwAGcALQB0AGUAbQBwAGwAYQB0AGUARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAII
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwECAQMBBAAJZmlsbC1zdGFyCmVtcHR5LXN
0YXIAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"],
[class*=" sf-icon-"] {
font-family: 'rating-template' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-fill-star:before {
content: "\e700";
}
.sf-icon-empty-star:before {
content: "\e701";
}
}

```

Using Emoji icon as rating symbol

You can use emojis of your choice as rating symbol by specifying them as template content within the `emptyTemplate` tag directive.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ emptyTemplate: '#emptyTemplate', value:
4, enableSingleSelection:true, enableAnimation:false });
// Render initialized Rating.

```

```
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>
  <script type="text/x-jsrender" id="emptyTemplate">
    ${if(index==0)}
    <span class='angry emoji'>&#128545;</span>${else if(index==1)}
    <span class='disagree emoji'>&#128577;</span>${else if(index==2)}
    <span class='neutral emoji'>&#128528;</span>${else if(index==3)}
    <span class='agree emoji'>&#128578;</span>${else}
    <span class='happy emoji'>&#128512;</span>
    ${/if}
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
```

```
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  margin: 50px auto;
  text-align: center;
}
/* To change the color of an unselected rating item. */
.e-rating-item-container:not(.e-rating-selected) .emoji {
  filter: grayscale(1);
}
```

Using SVG icon as rating symbol

You can use SVG icons of your choice as rating symbol by specifying them as template content within the `emptyTemplate` and `fullTemplate` tag directives.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ emptyTemplate:'#emptyTemplate',
fullTemplate:'#fullTemplate', value:4, enableAnimation:false });
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
    <script type="text/x-jsrender" id="emptyTemplate">
        <svg width="35" height="25" class="e-rating-svg-icon">
            <rect width="35" height="25" fill="transparent" style="stroke-
width:2;stroke:rgb(173,181,189)" />
        </svg>
    </script>
    <script type="text/x-jsrender" id="fullTemplate">
        <svg width="35" height="25" class="e-rating-svg-icon">
            <defs>
                <linearGradient id=grad${index} x1="0%" y1="0%" x2="100%"
y2="0%">
                    <stop class="start" offset="0%" />
                    <stop class="end" offset="100%" />
                </linearGradient>
            </defs>
            <rect width="35" height="25" fill="url(#grad${index})"
style="stroke-width:2;stroke:rgb(173,181,189)" />
        </svg>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
/* To change the size between items */

```

```
.e-rating-container .e-rating-item-container {
  padding: 0px;
}
/* To set the gradient color */
.e-rating-svg-icon #grad0 .start {
  stop-color: #FF0000;
}
.e-rating-svg-icon #grad0 .end,
.e-rating-svg-icon #grad1 .start {
  stop-color: #ff5101;
}
.e-rating-svg-icon #grad1 .end,
.e-rating-svg-icon #grad2 .start {
  stop-color: #ffc801;
}
.e-rating-svg-icon #grad2 .end,
.e-rating-svg-icon #grad3 .start {
  stop-color: #dbe300;
}
.e-rating-svg-icon #grad3 .end,
.e-rating-svg-icon #grad4 .start {
  stop-color: #8bc301;
}
.e-rating-svg-icon #grad4 .end {
  stop-color: #4eaa01;
}
```

Using PNG image as rating symbol

You can use PNG images of your choice as rating symbol by specifying them as template content within the `emptyTemplate` and `fullTemplate` tag directives.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ emptyTemplate: '#emptyTemplate',
fullTemplate: '#fullTemplate', value: 4 });
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
    <script type="text/x-jsrender" id="emptyTemplate">
        
    </script>

    <script type="text/x-jsrender" id="fullTemplate">
        
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c9f;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}

```

Events in EJ2 JavaScript Rating control

This section describes the rating events that will be triggered when appropriate actions are performed. The following events are available in the rating control.

[beforeItemRender](#)

The rating control triggers the [beforeItemRender](#) event before rendering each rating item. The [RatingItemEventArgs](#) passed as an event argument provides the details of the item to be rendered.

```
`js
// Initialize the Rating control.
var rating = new ej.inputs.Rating({
  beforeItemRender: (args)=> {
    //Your required action here
  }
});
// Render initialized Rating.
rating.appendTo('#rating');
```

[created](#)

The rating control triggers the [created](#) event when the rendering of the rating control is completed.

```
`js
// Initialize the Rating control.
var rating = new ej.inputs.Rating({
  created: ()=> {
    //Your required action here
  }
});
// Render initialized Rating.
rating.appendTo('#rating');
```

[onItemHover](#)

The rating control triggers the [onItemHover](#) event when the rating item is hovered. The [RatingHoverEventArgs](#) passed as an event argument provides the details of the hovered item.

```
`js
// Initialize the Rating control.
var rating = new ej.inputs.Rating({
  onItemHover: (args)=> {
```

```
//Your required action here
}
});
// Render initialized Rating.
rating.appendTo('#rating');
```

valueChanged

The rating control triggers the [valueChanged](#) event when the value of the rating is changed. The [RatingChangedEventArgs](#) passed as an event argument provides the details when value is changed.

```
`js
// Initialize the Rating control.
var rating = new ej.inputs.Rating({
valueChanged: (args)=> {
//Your required action here
}
});
// Render initialized Rating.
rating.appendTo('#rating');
```

Below example demonstrates the valueChanged event of the Rating control.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({
  valueChanged: (args)=> {
    alert("Previous Value:"+args.previousValue+"\nValue:"+args.value);
  }
});
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}

```

Appearance in EJ2 JavaScript Rating control

You can also customize the appearance of rating control.

Items count

You can specify the number of rating items using the [itemsCount](#) property.

INDEX.JS

```
ej.base.enableRipple(true);  
// Initialize the Rating control.  
var rating = new ej.inputs.Rating({itemsCount:8, value:3.0});  
// Render initialized Rating.  
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 Rating</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="TypeScript Rating Control">  
  <meta name="author" content="Syncfusion">  
  <link href="styles.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <div class="wrap">  
      <input id="rating">  
    </div>  
  </div>  
<script>  
var ele = document.getElementById('container');  
if(ele) {  
  ele.style.visibility = "visible";  
}  
</script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

STYLES.CSS

```
#container {  
  visibility: hidden;  
}  
#loader {  
  color: #008cff;  
  font-family: 'Helvetica Neue','calibiri';
```

```
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 50px auto;
text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
/* To change rating symbol border color */
-webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
/* To change rated symbol fill color and un-rated symbol fill color */
background: linear-gradient(to right, #ffe814 var(--rating-value),
#d8d7d4 var(--rating-value));
background-clip: text;
-webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
/* To change the size between items */
margin: 0px 7px;
}
@font-face {
font-family: 'custom-icon';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vSfQAAAEoAAAAVmntYXDnEudXAAABiAAAAADZnbHlmVIZ
rowAAACgAAAEYAGVhZCK6KOUAAADQAAAAANmhoZWEIUAQDAAAArAAAACRobXR4CAAAAAAAAYAAAA
IbG9jYQCMAAAAAAHAAAAABmlheHABDQCJAAABCAAAACBuYw1lv3dY+QAAAuAAAAJVCg9zdNl2Ynk
AAAU4AAAAALwABAAAEAAAAAFwEAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAAEGWKHv8
PPPUACwQAAAAAN/UcgcAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAAAAAAAAEAAAAACAH0AAQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAAQAAZAAABQAAAOcZAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
AAAACAAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAAEAAAA
AAAAAjAAAAAEAAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQ
EAWICAQEBAQIDAQWQBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGeBAQECAgMEBAUFBgYHCAgICQoKCgs
MDAwMDAwNDawNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
LDAwLCgsKCgkICQgHBwYFBQQDAwIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAgQ
EBQAAABIA3gABAAAAAAAAAAAAEAAAAABAAAAAABAAAsAAQABAAAAAAACAACADAABAAAAAADAAAsAEwA
BAAAAAAEAAsAHgABAAAAAAAFAAAsAKQABAAAAAAAGAAsANAABAAAAAAAKACwAPwABAAAAAALABIA
AwADAAEEECQAAAAIAfQADAAEEECQABABYAfwADAAEEECQACAA4AlQADAAEEECQADABYAowADAAEEECQA
EABYAuQADAAEEECQAFABYazwADAAEEECQAGABYA5QADAAEEECQAKAFgA+wADAAEEECQALACQBwBjdXN
0b20taWNvb1JlZ3VsYXJjdXN0b20taWNvbmlc3RvbS1pY29uVmVyc2lvbiAxLjBjdXN0b20taWN
vbkZvbnpGZ2VuzXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXN
pb24uY29tACAAAYwB1AHMAADABvAG0ALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAIYwB1AHMAADABvAG0
ALQBpAGMAbwBuAGMAQbZAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAUADAAAYwB1
AHMAADABvAG0ALQBpAGMAbwBuAEYAbwBuAHQAIAbnAGUAbgB1AHIAIYQB0AGUAZAAGAHUAcwBpAG4
AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwA
```

```

uAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAIBAgEDAAVoZWYdAAAAA==) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
    font-family: 'custom-icon' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
    content: "\e702";
}

```

Disabled

You can disable the rating control by using the [disabled](#) property. When the `disabled` property is set to `true`, the rating control will be disabled and the user will not be able to interact with it and a disabled rating control may have a different visual appearance than an enabled one.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3, disabled:true });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
    /* To change rating symbol border color */
    -webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
    /* To change rated symbol fill color and un-rated symbol fill color */
    background: linear-gradient(to right, #ffe814 var(--rating-value),
#d8d7d4 var(--rating-value));
    background-clip: text;
    -webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
    /* To change the size between items */
    margin: 0px 7px;
}

```



```

}
@font-face {
  font-family: 'custom-icon';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSfQAAAEoAAAAVmNtYXNlEudXAAABiAAAADZnbHlmVIZ
rowAAAcgAAAEYAGVhZCK6KOUAAADQAAANmhoZWEIUADAAARAAAAACRobXR4CAAAAAAAYAAAAA
IbG9jYQCMAAAAAHAAAAABmlheHABDQCJAAABCAAAACBuYW1lV3dY+QAAAUAAAAJVcG9zdN12Ynk
AAAU4AAAAALwABAAAEAAAAAFwEAAAAAAD8wABAAAAAAGABAAAAQAEEGwKhV8
PPPUACwQAAAAAN/UcgCAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAEEAAACAH0AAQAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
AAACAAAAAAwAAABQAAwABAAAAFAAEACIAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAEAAAA
AAAAAajAAAAEAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQ
EAWICAQEBAQIDAwQFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBAGQECAGMEBAUFBgYHCAgICQoKCgs
MDAwMDAwNDwNDwNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBAREERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
LDawLCgsKCgkICQgHBwYFBQQDAWIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAGQ
EBQAAABIA3gABAAAAAEEAAABAAAAAABAAASAAQABAAAAAACAACADAABAAAAAADAASAEwA
BAAAAAAEAASAHgABAAAAAFAASAKQABAAAAAAGAASANAABAAAAAAKACwAPwABAAAAAALABI
AawADAAEECQAAAAIAfQADAAEECQABABYafwADAAEECQACAA4AlQADAAEECQADABYAowADAAEECQA
EABYAUQADAAEECQAFABYAZwADAAEECQAGABYA5QADAAEECQAKAFgA+wADAAEECQALACQBUyBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbmNlc3RvbS1pY29uVmVyc2lvd3d3LnN5bmNmdXN
vbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAAdABvAG0ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyWb1AHMAAdABvAG0
ALQBpAGMABwBuAGMAdQBzAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB
1AHMAAdABvAG0ALQBpAGMABwBuAEYABwBuAHQAIAbnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4
AZwAgAFMAeQBwAGMAZgBlAHMAaQBvAG4AIAbnAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwA
uAHMAeQBwAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAa
AAAAAAAAAIBAgEDAAVoZWYdAAAAA==) format('true');
  font-weight: normal;
  font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
  font-family: 'custom-icon' !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
  content: "\e702";
}

```

Visible

You can use the [visible](#) property of the rating control to control the visibility of the control. When the **visible** property is set to **true**, the rating control will be visible on the page. When it is set to **false**, the control will be hidden.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.

```

```

var rating = new ej.inputs.Rating({ value:3, visible:true });
// Render initialized Rating.
rating.appendTo('#rating');
document.getElementById("btn").onclick = function () {
    rating.visible = (rating.visible) ? false : true;
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <button id="btn">Visible</button>
      <input id="rating">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
}

```

```

font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 50px auto;
text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
/* To change rating symbol border color */
-webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
/* To change rated symbol fill color and un-rated symbol fill color */
background: linear-gradient(to right, #ffe814 var(--rating-value),
#d8d7d4 var(--rating-value));
background-clip: text;
-webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
/* To change the size between items */
margin: 0px 7px;
}
@font-face {
font-family: 'custom-icon';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSfQAAAEoAAAAVmNtYXNlEudXAAABiAAAADZnbHlmVIZ
rowAAAcgAAAEYagVhZCK6KOUAAADQAAAAANmhoZWEIUAQDAAAArAAAACRobXR4CAAAAAAAYAAAAA
IbG9jYQCMAAAAAHAAAAABmlheHABDQCJAAABCAAAACBuYW1lv3dY+QAAAUAAAAJVcG9zdN12Ynk
AAAU4AAAAALwABAAAEAAAAAFwEAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAAEGWKv8
PPPUACwQAAAAAAN/UcgCAAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAAAAAAAAEAAAACAH0AAQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAA
AAAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAAEAAAA
AAAAAIAAAAAEAAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHbGyYFBQQ
EAWICAQEBAQIDAQWFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBQECAGMEBAUFBgYHCAgICQoKCgs
MDAwMDAwNDANDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ00DA0NDA0
LDAwLCgsKCgkICQgHBwYFBQQDAWIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAGQ
EBQAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAAsAAQABAAAAAAACAACADAABAAAAAADAAsAEwA
BAwAAAAAEAAAsAHgABAAAAAAFAAsAKQABAAAAAAAGAAAsANAABAAAAAAAKACwAPwABAAAAAALABI
AawADAAEECQAAAAIAfQADAAEECQABABYAfwADAAEECQACAA4AlQADAAEECQADABYAowADAAEECQA
EABYAQADAAEECQAFABYazwADAAEECQAGABYA5QADAAEECQAKAFga+wADAAEECQALACQBvYBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbmNlc3RvbSlpY29uVmVyc2lvbiAxLjBjdXN0b20taWN
vbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1ld3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAdABvAG0ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyBwB1AHMAdABvAG0
ALQBpAGMABwBuAGMAdQBzAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB
1AHMAdABvAG0ALQBpAGMABwBuAEYAbwBuAHQAIAbnAGUAbgB1AHIAyQB0AGUAZAAGAHUAcwBpAG4
AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAbnAGUAdABYAG8AIABTahQAdQBkAGkAbwB3AHcAdwA

```

```

uAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAIBAgEDAAVoZWYdAAAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
    font-family: 'custom-icon' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
    content: "\e702";
}

```

Read only

You can use the [readOnly](#) property of the rating control to make the control non-interactive and prevent the user from changing the rating value.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3, readOnly:true });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div class="wrap">
            <input id="rating">
        </div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
    /* To change rating symbol border color */
    -webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
    /* To change rated symbol fill color and un-rated symbol fill color */
    background: linear-gradient(to right, #ffe814 var(--rating-value),
    #d8d7d4 var(--rating-value));
    background-clip: text;
    -webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
    /* To change the size between items */
    margin: 0px 7px;
}
@font-face {

```

```
font-family: 'custom-icon';
src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMj1vSfQAAAEoAAAAVmNtYXNlEudXAAABiAAAADZnbHlmVIZrowAAACgAAAEYaGVhZCK6KOUAAADQAAAAANmhoZWEIUAQDAAAArAAAACRobXR4CAAAAAAAAYAAAAIbG9jYQCMAAAAAAHAAAAABmlheHABDQCJAAABCAAAACBuYW1l3dY+QAAAUAAAAJVcG9zdN12YnkAAAU4AAAAALwABAAAEAAAAAFwEAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAAEGWKhV8PPPUACwQAAAAAAN/UcgcAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAAAAAAAAEAAAACAH0AAQAAAAAAGAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAAEAAAAAAAJAAAAEAAAAA/MD2gB8AAATDxYVHW8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQEAWICAQEBAQIDAQwFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBAQECAGMEBAUFBgYHCAGICQoKCgsMDAwMDAwNDawNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQkKCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSiWk0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0LDAwLCgsKCgkICQgHBwYFBQQDAWIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAgQEBQAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAAAAABAAsAAQABAAAAAAAAACAacADAABAAAAAAAAADAAsAEwABAAAAAAAAEAAAsAHgABAAAAAAAFAAAsAKQABAAAAAAGAAsANAABAAAAAAAKACwAPwABAAAAAALABI AawADAAEECQAAAAIAfQADAAEECQABABYAfwADAAEECQACAA4AlQADAAEECQADABYAowADAAEECQAEABYAuwQADAAEECQAFABYAzwADAAEECQAGABYA5QADAAEECQAKAFgA+wADAAEECQALACQBUIYBjdXN0b20taWNvb1JlZ3VsYXJjdXN0b20taWNvbmlc3RvbSlpY29uVmVyc2lvd3d3LnN5bmNmdXNpb24uY29tACAAYwB1AHMAAdABvAG0ALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAyWb1AHMAAdABvAG0ALQBpAGMAbwBuAGMAAdQBzAHQABwbTAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAUaADAAYwB1AHMAAdABvAG0ALQBpAGMAbwBuAEYAbwBuAHQAIABnAGUAbgB1AHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAIABagEDAaVoZWfYdAAAAA==) format('truetype');
font-weight: normal;
font-style: normal;
}

.custom-icon .e-icons.e-star-filled {
font-family: 'custom-icon' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}

.custom-icon .e-icons.e-star-filled:before {
content: "\e702";
}
```

CssClass

You can customize the appearance of the rating control, such as by changing its colors, fonts, sizes, or other visual aspects by using the `cssClass` property.

Changing rating symbol border color

You can change the rating icon border color in rating control, you can use the `cssClass` property and set the `text-stroke` CSS property of `.e-rating-icon` to your desired border color.

INDEX.JS

```
ej.base.enableRipple(true);
```

```
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3, cssClass:"custom-border" });
// Render initialized Rating.
rating.appendTo('#rating');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
```

```

position: absolute;
top: 45%;
width: 30%;
}
.wrap {
margin: 50px auto;
text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
/* To change rating symbol border color */
-webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
/* To change rated symbol fill color and un-rated symbol fill color */
background: linear-gradient(to right, #ffe814 var(--rating-value),
#d8d7d4 var(--rating-value));
background-clip: text;
-webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
/* To change the size between items */
margin: 0px 7px;
}
@font-face {
font-family: 'custom-icon';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMjlvSfQAAAEoAAAAVmNtYXDNtEudXAAABiAAAADZnbHlmVIZ
rowAAAcgAAAEYAGVhZCK6KOUAADQAAAAANmhoZWEIUADAAAArAAAACRobXR4CAAAAAAAYAAAA
IbG9jYQCMAAAAAHAAAAABmlheHABDQCJAAABCAAAACBuYwllv3dY+QAAAUAAAAJVCg9zdNl2Ynk
AAAU4AAAAALwABAAAEAAAAAFwEAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAEEGwKhV8
PPPUACwQAAAAAN/UcgAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAAAAAAAAEAAAACAH0AAQAAAA
AAgAAAAoAcgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAA
AAAACAAAAAwAAABQAawABAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAAEAAAA
AAAAAjaAAAAEAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQ
EAwICAQEBAQIDAwQFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBAGQECAGMEBAUFBgYHCAgICQoKCGs
MDAwMDAwNDawNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQWNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
LDAwLCgsKCgkICQgHBwYFBQQDAwIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQEACgQ
EBQAAABIA3gABAAAAAAAAAAAAEAAAAABAAAAAABAAAsAAQABAAAAAAACAAcADAABAAAAAADAAsAEwA
BAAAAAAEAAsAHgABAAAAAAFAAsAKQABAAAAAAAGAAsANAABAAAAAAAKACwAPwABAAAAAALABI
AawADAAEECQAAAAIAfQADAAEECQABABYafwADAAEECQACAA4ALQADAAEECQADABYAowADAAEECQA
EABYAuwADAAEECQAFABYazwADAAEECQAGABYA5QADAAEECQAKAFgA+wADAAEECQALACQBUyBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbmNlc3RvbS1pY29uVmVyc2lubiAxLjBjdXN0b20taWN
vbkZvbG9zZ2VuzXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAAdABvAG0ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAIAYwB1AHMAAdABvAG0
ALQBpAGMABwBuAGMAdQBzAHQAbwBtAC0AAQBJAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB1
AHMAAdABvAG0ALQBpAGMABwBuAEYABwBuAHQAIAbnAGUAbgB1AHIAIYQB0AGUAZAAGAHUAcwBpAG4
AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwA
uAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAA
AAAAAAAAAAAIABgEDAAVoZWfydAAAAA==) format('true-type');
font-weight: normal;
font-style: normal;

```



```

}
.custom-icon .e-icons.e-star-filled {
  font-family: 'custom-icon' !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
  content: "\e702";
}

```

Changing rated/un-rated symbol fill color

You can customize the fill colors of rated and un-rated icons in the rating control using the `cssClass` property and the `linear-gradient` color-stops in the `background` CSS property of `.e-rating-icon`. The **first** color-stop defines the rated fill color and the **second** defines the un-rated fill color.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3, cssClass:"custom-fill" });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```
<div id="container">
  <div class="wrap">
    <input id="rating">
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008c9f;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
    /* To change rating symbol border color */
    -webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
    /* To change rated symbol fill color and un-rated symbol fill color */
    background: linear-gradient(to right, #ffe814 var(--rating-value),
#d8d7d4 var(--rating-value));
    background-clip: text;
    -webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
    /* To change the size between items */
    margin: 0px 7px;
}
}
@font-face {
    font-family: 'custom-icon';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAAKAIAAAAwAgTlMvMjlvSfOAAAEoAAAAVmNtYXDNdEudXAAABiAAAAADZnbHlMViZ
```

```

rowAAAcgAAAEYAGVhZCK6KOUAADQAAAAANmhoZWEIUAQDAAAArAAAACRobXR4CAAAAAAAYAAAA
IbG9jYQCMAAAAAHHAAAAABmlheHABDQCJAAABCAAAACBuYW1lV3dY+QAAAUAAAAJVcG9zdN12Ynk
AAAU4AAAAALwABAAAEAAAAAFWEAAAAAAD8wABAAAAAABAAAAAABAAAAAABAAAAAQAAGWKv8
PPPUACwQAAAAAN/UcgCAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAABAAAAACAH0AAQAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAA
AAAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAEAAAA
AAAAAJAAAAEAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHbGyYFBQQ
EAWICAQEBAQIDAwQFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBQECAGMEBAUFBgYHCAGICQoKCgs
MDAwMDAwNDawNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
LDAwLCgsKCgkICQgHbWYFBQQDAwIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAGQ
EBQAAABIA3gABAAAAAABAAAAEAAAAAABAAAAAABAAAsAAQABAAAAAACAacADAABAAAAAADAAsAEwA
BAAAAAAEAAAsAHgABAAAAAABFAAsAKQABAAAAAABGAAsANAABAAAAAABKACwAPwABAAAAAABLABI
AawADAAEECQAAAAIAfQADAAEECQABABYAfWADAAEECQACAA4AlQADAAEECQADABYAowADAAEECQA
EABYAUQADAAEECQAFABYAZwADAAEECQAGABYA5QADAAEECQAKAFgA+wADAAEECQALACQBUyBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbmNlc3RvbS1pY29uVmVyc2lvbiAxLjBjdXN0b20taWN
vbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAdABvAG0ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyWb1AHMAdABvAG0
ALQBpAGMABwBuAGMAdQBzAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB
1AHMAdABvAG0ALQBpAGMABwBuAEYAbwBuAHQAIABnAGUAbgB1AHIAyQB0AGUAZAAGAHUAcwBpAG4
AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwA
uAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAABAAAAAABAAAAAABAAAAA
AAAAAAAAAIBAgEDAAVoZWZydAAAAA==) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
    font-family: 'custom-icon' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
    content: "\e702";
}

```

This will customize the rated fill color to `#ffe814` and un-rated fill color to `#d8d7d4`. `--rating-value` in the linear-gradient provides the current value of the rating item.

Changing the item spacing

You can change the space between the rating items in rating control, by using the `cssClass` property and setting the `margin` / `padding` CSS property of `.e-rating-item-container` to your desired size.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value:3, cssClass:"custom-font" });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008c8f;
  font-family: 'Helvetica Neue', 'calibri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
```

```

.wrap {
  margin: 50px auto;
  text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
  /* To change rating symbol border color */
  -webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
  /* To change rated symbol fill color and un-rated symbol fill color */
  background: linear-gradient(to right, #ffe814 var(--rating-value),
#d8d7d4 var(--rating-value));
  background-clip: text;
  -webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
  /* To change the size between items */
  margin: 0px 7px;
}
@font-face {
  font-family: 'custom-icon';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSfQAAAEoAAAAVmNtYXNlEudXAAABiAAAADZnBHlmVIZ
rowAAAcgAAAEYaGVhZCK6KOUAAADQAAANmhoZWEIUaQDAAARAAAACRobXR4CAAAAAAAYAAAA
IbG9jYQCMAAAAAHAAAAABmlheHABDQCJAAABCAAAACBuYW1lv3dY+QAAAUAAAAJVCg9zdN12Ynk
AAAAUAAAAALwABAAAEAAAAAFwEAAAAAAD8wABAAAAAIAAAAGABAAAAQAEGWKhV8
PPPUACwQAAAAAN/UcgcAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAIAAAACAH0AAQAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZABQAAaokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAEAAAA
AAAAAJAAAAEAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQ
EAWICAQEBAQIDAQWFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBQECAGMEBAUFBgYHCAGICQoKCgs
MDAwMDAwNDawNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQWNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
LDAwLCgsKCgkICQgHBwYFBQQDAWIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAGQ
EBQAAABIA3gABAAAAAIAAAEAAAAAABAAAAAABAASAAQABAAAAAACAADAABAAAAAADAASAEWA
BAAAAAAEAAshgABAAAAAAFAAsAKQABAAAAAAGAAsANAABAAAAAAKACwAPwABAAAAAALABI
AawADAAEEECQAAAAIAfQADAAEEECQABABYafwADAAEEECQACAA4AlQADAAEEECQADABYAowADAAEEECQA
EABYAUQADAAEEECQAFABYazwADAAEEECQAGABYA5QADAAEEECQAKAFgA+wADAAEEECQALACQBUyBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbmNlc3RvbSlpY29uVmVyc2lvbiAxLjBjdXN0b20taWN
vbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAAdABvAG0ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyBwB1AHMAAdABvAG0
ALQBpAGMABwBuAGMAAdQBzAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB
1AHMAAdABvAG0ALQBpAGMABwBuAEYAbwBuAHQAIAbnAGUAbgB1AHIAyQB0AGUAZAAGAHUAcwBpAG4
AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwA
uAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAIAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAIBAgEDAAVoZWYdAAAAA==) format('trueType');
  font-weight: normal;
  font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
  font-family: 'custom-icon' !important;
  speak: none;
}

```

```

font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
  content: "\e702";
}

```

Changing icon using CssClass

You can change the rating item icon in rating control, you can use the `cssClass` property and set the `content` CSS property of `.e-icons.e-star-filled:before` to your desired font icon.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Rating control.
var rating = new ej.inputs.Rating({ value: 3, cssClass: "custom-icon" });
// Render initialized Rating.
rating.appendTo('#rating');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Rating</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Rating Control">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="wrap">
      <input id="rating">
    </div>
  </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    margin: 50px auto;
    text-align: center;
}
.e-rating-container.custom-border .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-border .e-rating-item-container .e-rating-icon {
    /* To change rating symbol border color */
    -webkit-text-stroke: 2px #ae9e9d;
}
.e-rating-container.custom-fill .e-rating-item-list:hover .e-rating-item-
container .e-rating-icon,
.e-rating-container.custom-fill .e-rating-item-container .e-rating-icon {
    /* To change rated symbol fill color and un-rated symbol fill color */
    background: linear-gradient(to right, #ffe814 var(--rating-value),
#d8d7d4 var(--rating-value));
    background-clip: text;
    -webkit-background-clip: text;
}
.e-rating-container.custom-font .e-rating-item-container {
    /* To change the size between items */
    margin: 0px 7px;
}
@font-face {
    font-family: 'custom-icon';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vSfQAAAEoAAAAVmNtYXDNdEudXAAABiAAAADZnbHlmVIZ
rowAAACgAAAEYAGVhZCK6KOUAAADQAAAAANmhoZWEIUQAADAAARAAAAACRobXR4CAAAAAAAYAAAAA
IbG9jYQCMAAAAAAHAAAAABmlheHABDQCJAAABCAAAACBuYW11v3dY+QAAAUAAAAAJVcG9zdN12Ynk
AAAU4AAAAALwABAAAEAAAAAFwEAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAAEGWKhV8
PPPUACwQAAAAAAN/UcgcAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAAAAAAAAEAAAAACAH0AAQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
```

```

AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAEAAAA
AAAAAJAAAAEAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQ
EAWICAQEBAQIDAQFBBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBAGECAGMEBAUFBgYHCAgICQoKCgs
MDAwMDAwNDawNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
LDAwLCgsKCgkICQgHBwYFBQQDAWIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQEACgQ
EBQAAABIA3gABAAAAAEEEEAAAAABAAAAAABAAsAAQABAAAAAACAACADAABAAAAAADAAsAEwA
BAAAAAAEAAsAHgABAAAAAFAAsAKQABAAAAAAGAAsANAABAAAAAAKACwAPwABAAAAAALABI
AawADAAEECQAAAAIAfQADAAEECQABABYafwADAAEECQACAA4AlQADAAEECQADABYAowADAAEECQA
EABYAuQADAAEECQAFABYazwADAAEECQAGABYA5QADAAEECQAKAFgA+wADAAEECQALACQBUyBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbmNlc3RvbS1pY29uVmVyc2lubiAxLjBjdXN0b20taWN
vbkZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZG1vd3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAdABvAG0ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyWB1AHMAdABvAG0
ALQBpAGMABwBuAGMAdQBzAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB
1AHMAdABvAG0ALQBpAGMABwBuAEYAbwBuAHQAIABnAGUAbgB1AHIAyQB0AGUAZAAGAHUAcwBpAG4
AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwA
uAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAIABgEDAAVoZWZydAAAAA==) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
    font-family: 'custom-icon' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.custom-icon .e-icons.e-star-filled:before {
    content: "\e702";
}

```

Accessibility in EJ2 JavaScript Rating control

Accessibility is achieved in the rating control through **WAI-ARIA** standard and keyboard navigations. The rating control can be effectively accessed through assistive technologies such as screen readers.

Keyboard interaction

The rating control is interactive with below keyboard shortcuts.

| Keyboard shortcuts | Actions |

|-----|-----|

| Space | If a **Reset Button** is focused, resets the value to min value. |

| Arrow Up | Increases the value. |

| Arrow Left | Decreases the value and in RTL mode, increases the value. |

| Arrow Down | Decreases the value. |

| Arrow Right | Increases the value and in RTL mode, decreases the value. |

ARIA attribute

The following ARIA attributes are used in rating control based on its state.

| Properties | Functionality |

| ----- | ----- |

| role | This attribute is added to the div element to describe the actual role. |

| aria-label | Attribute provides the text label with some default description for the Rating and its items. |

| aria-valuemin | It defines the minimum value of rating. |

| aria-valuemax | It defines the maximum value of rating. |

| aria-valuenow | It defines the current value of rating. |

Ribbon

Tabs and Groups

The Ribbon control consists of a series of tabs that are organized into groups to enable quick access to specific commands or tools. Each tab contains a set of groups, and each group contains collections of items that are logically related to each other.

Adding Tabs

You can use the [tabs](#) property to add tabs to the Ribbon control and define the content of the tab header by using the [header](#) property.

INDEX.JS

```
var ribbon = new ej.ribbon.Ribbon({
  tabs: [
    {
      header: "Home"
    },
    {
      header: "Insert"
    }
  ]
});
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding Groups

You can use the [groups](#) property to add groups for each tab in the Ribbon and define the name of the group header by using the [header](#) property.

INDEX.JS

```

var ribbon = new ej.ribbon.Ribbon({
  tabs: [
    {
      header: "Home",
      groups: [
        {
          header: "Clipboard"
        }
      ]
    }
  ],
});

```

```

    {
      header: "Insert",
      groups: [
        {
          header: "Tables"
        }
      ]
    }
  ]
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ribbon"></div>
  </div>
<script>
var ele = document.getElementById('container');

```

```
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Adding Items

You can add collections of items to each group by using the [collections](#) and [items](#) properties.

INDEX.JS

```
var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                }
              },
            ],
          },
        ],
      },
      {
        items: [
          {
            type: "Button",
            buttonSettings: {
              content: "Cut",
              iconCss: "e-icons e-cut",
            }
          },
          {
            type: "Button",
            buttonSettings: {
              content: "Copy",
              iconCss: "e-icons e-copy",
            }
          },
        ],
      },
    ],
  }
];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ribbon"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

For more information on the built-in and how to add custom Ribbon items, you can visit the [items](#) page.

Ribbon Items

Ribbon renders various built-in items based on the item [type](#) property. By default, the type property is set as **Button** which renders the Button.

Built-in items

You can render the built-in Ribbon items by using the [items](#) property, to specify the [type](#) property.

The following table explains the built-in items and their actions.

Built-in Ribbon Items	Actions
Button	Renders button as ribbon item.
CheckBox	Renders checkbox as ribbon item.
DropDown	Renders dropdownbutton as ribbon item.
SplitButton	Renders splitbutton as ribbon item.
ComboBox	Renders combobox as ribbon item.
ColorPicker	Renders color picker as ribbon item.
GroupButton	Renders groupbutton as ribbon item.

Button items

You can render the built-in button Ribbon item by setting the [type](#) property as `Button` you can render a Button item. You can also customize the button item using the [RibbonButtonSettingsModel](#), which provides options such as `iconCss`, `content`, `isToggle` and more.

Toggle button

The [isToggle](#) property can be used to define whether the button act as a toggle button or not. By default, the value is `false`.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut",
            isToggle: true
          }
        }]
      }
    ]
  }]
}]
```

```

    }}
  });
  var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs
  });
  ribbon.appendTo("#ribbon");
  ,

```

Checkbox items

You can render the built-in checkBox Ribbon item by setting the [type](#) property to `CheckBox`. You can also customize the checkBox item using the [RibbonCheckBoxSettingsModel](#), which provides options such as `labelPosition`, `label`, `checked` and more.

Checkbox state

You can use the [checked](#) property to handle the checked or unchecked state. By default, the value is `false`.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "View",
    collections: [
      {
        items: [{
          type: "CheckBox",
          checkBoxSettings: {
            label: "Ruler",
            checked: true
          }
        }
      ]
    ]
  }
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
,

```

Defining label

You can use the [label](#) property to add a caption for the CheckBox. The label position can be set **Before** or **After**, by using the [labelPosition](#) property. By default, the labelPosition is **After**.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "View",
    collections: [
      {
        items: [{
          type: "CheckBox",
          checkBoxSettings: {
            label: "Ruler",
            labelPosition: "Before"
          }
        }
      ]
    }
  ]
}];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

DropDown button items

You can render the built-in dropDown Ribbon item by setting the [type](#) property to **DropDown**. You can also customize the dropDown item through [RibbonDropDownSettingsModel](#), which provides options such as **iconCss**, **content**, **target** and more.

Target

The [target](#) property specifies the element selector to be displayed in the DropDownButton popup.

INDEX.JS

```
var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Illustrations",
        collections: [
          {
```



```

        items: [
            {
                type: "DropDown",
                dropDownSettings: {
                    content: "Picture",
                    iconCss: "e-icons e-image",
                    target: "#pictureList"
                }
            },
        ],
    },
]
}
}
]
}
];
var list = new ej.lists.ListView({
    headerTitle: 'Insert Picture From',
    dataSource: ['Stock Images', 'This device', 'Online Images'],
    showHeader: true
});
list.appendTo('#pictureList');
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">

```

```

<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
        <div id="pictureList"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Split button items

You can render the built-in splitButton Ribbon item by setting the [type](#) property to `SplitButton`. You can also customize the splitButton item through [RibbonSplitButtonSettingsModel](#), which provides options such as `iconCss`, `items`, `target` and more.

Target

The [target](#) property specifies the element selector to be displayed in the SplitButton popup.

INDEX.JS

```

var tabs = [
    {
        header: "Home",
        groups: [
            {
                header: "Illustrations",
                collections: [
                    {
                        items: [
                            {
                                type: "SplitButton",
                                splitButtonSettings: {
                                    content: "Picture",
                                    iconCss: "e-icons e-image",
                                    target: "#pictureList"
                                }
                            }
                        ],
                    },
                ],
            }
        ]
    }
]

```

```

];
var list = new ej.lists.ListView({
  headerTitle: 'Insert Picture From',
  dataSource: ['Stock Images', 'This device', 'Online Images'],
  showHeader: true
});
list.appendTo('#pictureList');
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ribbon"></div>
    <div id="pictureList"></div>
  </div>
<script>

```

```
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Combobox items

You can render the built-in comboBox Ribbon item by setting the `type` property to `ComboBox`. You can also customize the comboBox item through [RibbonComboBoxSettingsModel](#), which provides options such as `allowFiltering`, `autoFill`, `index`, `sortOrder` and more.

Filtering

You can use the [allowFiltering](#) property to filter the data items. The filtering operation is initiated automatically, as soon as you start typing characters. If no match is found the value of the `noRecordsTemplate` property will be displayed. By default, the value is `false`.

```
`js
```

```
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia"];
```

```
var tabs = [{
```

```
  header: "Home",
```

```
  groups: [{
```

```
    header: "Font",
```

```
    collections: [
```

```
    {
```

```
      items: [{
```

```
        type: "ComboBox",
```

```
        comboBoxSettings: {
```

```
          dataSource: fontStyle,
```

```
          index: 3,
```

```
          allowFiltering: true
```

```
        }
```

```
      ]
```

```
    }]
```

```
  }]
```

```
});
```

```
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
```

```
ribbon.appendTo("#ribbon");
```

Index

You can use the [index](#) property to get or set the selected item in the combobox.

```
`js
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara",
"Georgia"];
var tabs = [{
header: "Home",
groups: [{
header: "Font",
collections: [
{
items: [{
type: "ComboBox",
comboBoxSettings: {
dataSource: fontStyle,
index: 3
}
}]
}]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
```

SortOrder

You can use the [sortOrder](#) property to specify the order in which the DataSource should be sorted.

None	The data source is not sorted.
Ascending	The data source is sorted in ascending order.
Descending	The data source is sorted in descending order.

INDEX.JS

```
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math",
"Courier New", "Candara", "Georgia"];
```

```

var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ComboBox",
          comboBoxSettings: {
            dataSource: fontStyle,
            sortOrder: "Descending"
          }
        }]
      }
    ]
  }]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

  <div id="container">
    <div id="ribbon"></div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Colorpicker items

You can render the built-in colorPicker Ribbon item by setting the [type](#) property to `ColorPicker`. You can also customize the colorPicker item through [RibbonColorPickerSettingsModel](#), which provides options such as `value`, `columns`, `showButtons` and more.

Value

You can use the [value](#) property to specify the color value. The value should be specified as Hex code.

INDEX.JS

```
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ColorPicker",
          colorPickerSettings: {
            value: "#123456"
          }
        }]
      }
    ]
  }]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Groupbutton items

You can render the built-in groupbutton Ribbon item by setting the [type](#) property to `GroupButton`. You can also customize the groupbutton item using the [RibbonGroupButtonSettingsModel](#), which provides options such as `selection` and `items`.

Items

You can render the groupbutton items by using [items](#) property. You can also customize the groupbutton items through [RibbonGroupButtonItemModel](#), which provides options such as `content`, `iconCss`, `selected` and more.

Item content

You can use the [content](#) property to define the text content for the groupbutton.

```
`js
```

```
var tabs = [{
```



```
header: "Home",
groups: [{
  header: "Paragraph",
  collections: [
    {
      items: [{
        type: "GroupButton",
        allowedSizes: ej.ribbon.RibbonItemSize.Small,
        groupButtonSettings: {
          items: [{
            iconCss: 'e-icons e-align-left',
            content: 'Align Left'
          },
          {
            iconCss: 'e-icons e-align-center',
            content: 'Align Center'
          },
          {
            iconCss: 'e-icons e-align-right',
            content: 'Align Right'
          },
          {
            iconCss: 'e-icons e-justify',
            content: 'Justify'
          }
        ]
      }
    }
  ]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

Icon only

You can use the [iconCss](#) property to customize the groupbutton icon. If the `iconCss` property is not defined, the groupbutton will not be rendered.

INDEX.JS

```
var tabs = [{
  header: "Home",
  groups: [{
    header: "Paragraph",
    collections: [
      {
        items: [{
          type: "GroupButton",
          allowedSizes: ej.ribbon.RibbonItemSize.Small,
          groupButtonSettings: {
            items: [{
              iconCss: 'e-icons e-align-left'
            },
            {
              iconCss: 'e-icons e-align-center'
            },
            {
              iconCss: 'e-icons e-align-right'
            },
            {
              iconCss: 'e-icons e-justify'
            }
          ]
        }
      ]
    ]
  }
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-
scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="ribbon"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Selection

You can use the [selected](#) property to select the groupbutton item initially. When set to `true`, the button will be selected. By default the `selected` property is false.

INDEX.JS

```

var tabs = [{
  header: "Home",
  groups: [{
    header: "Paragraph",
    collections: [
      {
        items: [{
          type: "GroupButton",
          allowedSizes: ej.ribbon.RibbonItemSize.Small,
          groupButtonSettings: {
            items: [{
              iconCss: 'e-icons e-align-left',
              content: 'Align Left'
            },
            {
              iconCss: 'e-icons e-align-center',
              content: 'Align Center',
              selected: true
            },
            {
              iconCss: 'e-icons e-align-right',

```

```

        content: 'Align Right'
      },
      {
        iconCss: 'e-icons e-justify',
        content: 'Justify'
      }
    ]
  })
}

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-
scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="ribbon"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Single selection

You can set the [selection](#) property value as `RibbonGroupButtonSelection.Single` to make one selection at a time. It automatically deselects the previous choice when a different item is clicked.

INDEX.JS

```

var tabs = [{
  header: "Home",
  groups: [{
    header: "Paragraph",
    collections: [
      {
        items: [{
          type: "GroupButton",
          allowedSizes: ej.ribbon.RibbonItemSize.Small,
          groupButtonSettings: {
            selection: ej.ribbon.RibbonGroupButtonSelection.Single,
            items: [{
              iconCss: 'e-icons e-align-left',
              content: 'Align Left'
            },
            {
              iconCss: 'e-icons e-align-center',
              content: 'Align Center',
              selected: true
            },
            {
              iconCss: 'e-icons e-align-right',
              content: 'Align Right'
            },
            {
              iconCss: 'e-icons e-justify',
              content: 'Justify'
            }
          ]
        }
      ]
    }
  ]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0, user-
scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="ribbon"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple selection

You can set the [selection](#) property value as `RibbonGroupButtonSelection.Multiple` to select more than one button at a time. Users can select a button one by one to select multiple buttons.

INDEX.JS

```

var tabs = [{
  header: "Home",
  groups: [{
    header: "Paragraph",
    collections: [
      {
        items: [{
          type: "GroupButton",
          allowedSizes: ej.ribbon.RibbonItemSize.Small,

```

```

        groupButtonSettings: {
            selection:
ej.ribbon.RibbonGroupButtonSelection.Multiple,
            items: [{
                iconCss: 'e-icons e-bold',
                content: 'Bold'
            },
            {
                iconCss: 'e-icons e-italic',
                content: 'Italic',
                selected: true
            },
            {
                iconCss: 'e-icons e-underline',
                content: 'Underline'
            },
            {
                iconCss: 'e-icons e-strikethrough',
                selected: true,
                content: 'Strikethrough'
            }
        ]
    }
}
}

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-
scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="container">
<div id="ribbon"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Groupbutton in simplified mode layout

In simplified mode, the groupbutton will be rendered as a dropdownbutton. The dropdownbutton icon will be updated based on the button item selected. The initial button icon will be the set, if none of the buttons are selected.

INDEX.JS

```

var tabs = [{
  header: "Home",
  groups: [{
    header: "Paragraph",
    collections: [
      {
        items: [{
          type: "GroupButton",
          allowedSizes: ej.ribbon.RibbonItemSize.Small,
          groupButtonSettings: {
            selection: ej.ribbon.RibbonGroupButtonSelection.Single,
            items: [{
              iconCss: 'e-icons e-align-left',
              content: 'Align Left'
            },
            {
              iconCss: 'e-icons e-align-center',
              content: 'Align Center',
              selected: true
            },
            {
              iconCss: 'e-icons e-align-right',
              content: 'Align Right'
            },
            {
              iconCss: 'e-icons e-justify',
              content: 'Justify'
            }
          ]
        }
      ]
    }
  ]
}

```



```

    }
    }]
  }]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs, activeLayout: 'Simplified'
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, user-
  scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
  <div id="container">
  <div id="ribbon"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
  ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom items

You can customize the ribbon items with non-built-in items or HTML content by setting the [type](#) property to `Template`. This provides an option to customize the ribbon items with greater flexibility.

INDEX.JS

```
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "Template",
          itemTemplate: "#itemTemplate"
        }]
      }
    ]
  }]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  ribbon/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
```

```

<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
    <script type="text/x-jsrender" id="itemTemplate">
        <span class="ribbonTemplate ${activeSize}">
            <span class="e-icons e-video"></span>
            <span class="text">Video</span>
        </span>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.ribbonTemplate {
    display: flex;
    align-items: center;
    justify-content: center;
    cursor: pointer;
}
.ribbonTemplate.Large {
    flex-direction: column;
}
.ribbonTemplate.Large .e-icons {
    font-size: 35px;
}
.ribbonTemplate.Medium .e-icons,
.ribbonTemplate.Small .e-icons{
    font-size: 20px;
    margin: 15px 5px;
}
.ribbonTemplate.Small .text {
    display:none;
}

```

Items display Mode

You can use the [displayOptions](#) property to display the items in the Ribbon layout.

Auto	The items are displayed in all layouts based on the ribbon's overflow state.
Classic	The items are displayed only in the classic layout group.
Simplified	The items are displayed only in the simplified layout group.
Overflow	The items are displayed only in the overflow popup.

Display items in Classic only

To display the items only in the classic layout group, set the mode as `DisplayMode.Classic` in the [displayOptions](#) property.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          },
          displayOptions: ej.ribbon.DisplayMode.Classic
        }
      ]
    ]
  }
  ]
  });
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

Display items in Simplified only

To display the items only in the simplified layout group, set the mode as `DisplayMode.Simplified` in the [displayOptions](#) property.

```
`js
var tabs = [{
```

```

header: "Home",
groups: [{
  header: "Font",
  collections: [
    {
      items: [{
        type: "Button",
        buttonSettings: {
          content: "Cut",
          iconCss: "e-icons e-cut"
        },
        displayOptions: ej.ribbon.DisplayMode.Simplified
      }]
    }
  ]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

Display items in Overflow popup only

To display the items only in the overflow, set the mode as `DisplayMode.Overflow` in the [displayOptions](#) property.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }]
      }
    ]
  }
}];

```

```

},
displayOptions: ej.ribbon.DisplayMode.Overflow
}]
}]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

Enable or disable items

You can use the [disabled](#) property to disable a Ribbon item. It prevents the user interaction when set to **true**. By default, the value is **false**.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          disabled: true,
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        },
        {
          type: "CheckBox",
          disabled: true,
          checkBoxSettings: {
            label: "Ruler",
            checked: true
          }
        }
      ]
    }
  }
]
}
]
`

```

```

},
{
  type: "DropDown",
  disabled: true,
  dropDownSettings: {
    content: "Table",
    iconCss: "e-icons e-table"
  }
},
{
  type: "SplitButton",
  disabled: true,
  splitButtonSettings: {
    content: "Table",
    iconCss: "e-icons e-table"
  }
}
}
}
}
});
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");

```

Ribbon Layouts

The Ribbon allows to customize the layout by using the [activeLayout](#) property. The Ribbon component supports the following layouts:

Classic layout

In classic layout, the Ribbon component organizes the items and groups in a traditional form by setting the [activeLayout](#) property to [Classic](#). By default, the Ribbon component renders in the [Classic](#) layout.

INDEX.JS

```

var tabs = [
  {
    header: "Home",

```

```

groups: [
  {
    header: "Clipboard",
    collections: [
      {
        items: [
          {
            type: "Button",
            buttonSettings: {
              content: "Paste",
              iconCss: "e-icons e-paste",
            }
          },
        ],
      },
      {
        items: [
          {
            type: "Button",
            buttonSettings: {
              content: "Cut",
              iconCss: "e-icons e-cut",
            }
          },
          {
            type: "Button",
            buttonSettings: {
              content: "Copy",
              iconCss: "e-icons e-copy",
            }
          },
        ],
      },
    ],
  },
],
{
  header: "Insert",
  groups: [
    {
      header: "Tables",
      collections: [{
        items: [{
          type: "DropDown",
          dropDownSettings: {
            iconCss: "e-icons e-table",
            content: "Table",
            items: [
              { text: "Insert Table" }, { text: "Draw Table" },
              { text: "Convert Table" }, { text: "Excel
SpreadSheet" }
            ]
          }
        }
      ]
    }
  ]
}

```



```

    }
  ]
}
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  activeLayout: "Classic"
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ribbon"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```

```

    }
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Defining items size

You can use the [allowedSizes](#) property to set the allowed size for an item. The Ribbon items can be appeared in three different sizes: Large(large icon with text), Medium(small icon with text) and Small(small icon only). On resizing, the items size can be changed based on the available width of the tab content from the order of Large-> Medium-> Small and viceversa.

INDEX.JS

```

var tabs = [{
  header: "Home",
  groups: [{
    collections: [{
      items: [{
        type: 'SplitButton',
        allowedSizes: ej.ribbon.RibbonItemSize.Large,
        splitButtonSettings: {
          iconCss: 'e-icons e-paste',
          content: 'Paste',
          items: [{ text: 'Keep Source Format' }, { text: 'Merge format' },
            { text: 'Keep text only' }]
        }
      }
    ]
  }, {
    items: [{
      type: 'Button',
      allowedSizes: ej.ribbon.RibbonItemSize.Medium,
      buttonSettings: {
        content: 'Cut',
        iconCss: 'e-icons e-cut'
      }
    }, {
      type: 'Button',
      allowedSizes: ej.ribbon.RibbonItemSize.Small,
      buttonSettings: {
        content: 'Copy',
        iconCss: 'e-icons e-copy'
      }
    }
  ]
}
]
}];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Defining items orientation

The Ribbon group [orientation](#) property allows to manage how the items are aligned either in a **Row** or **Column**. By default, the orientation is set to **Column**, in which the items are arranged vertically.

INDEX.JS

```

var fontSize = ['8', '9', '10', '11', '12', '14', '16', '18', '20', '22',
'24', '26', '28', '36', '48', '72', '96'];

```

```

var fontStyle = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math',
'Courier New', 'Candara', 'Georgia', 'Impact', 'Segoe Print', 'Segoe
Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana', 'Windings'
];
var tabs = [{
  header: "Home",
  groups: [{
    collections: [{
      items: [{
        type: 'SplitButton',
        allowedSizes: ej.ribbon.RibbonItemSize.Large,
        splitButtonSettings: {
          iconCss: 'e-icons e-paste',
          content: 'Paste',
          items: [{ text: 'Keep Source Format' }, { text: 'Merge
format' }, { text: 'Keep text only' }]
        }
      }
    ]
  }, {
    items: [{
      type: 'Button',
      buttonSettings: {
        content: 'Cut',
        iconCss: 'e-icons e-cut'
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Copy',
        iconCss: 'e-icons e-copy'
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Format Painter',
        iconCss: 'e-icons e-format-painter'
      }
    }
  ]
}
], {
  orientation: 'Row',
  collections: [{
    items: [{
      type: 'ComboBox',
      comboBoxSettings: {
        dataSource: fontStyle,
        index: 3,
        allowFiltering: true,
        width: '150px'
      }
    }, {
      type: 'ComboBox',
      comboBoxSettings: {
        dataSource: fontSize,
        index: 3,
        width: '65px'
      }
    }
  ]
}
]
}
]

```

```

    }, {
      items: [{
        type: 'Button',
        buttonSettings: {
          content: 'Bold',
          iconCss: 'e-icons e-bold'
        }
      }, {
        type: 'Button',
        buttonSettings: {
          content: 'Italic',
          iconCss: 'e-icons e-italic'
        }
      }, {
        type: 'Button',
        buttonSettings: {
          content: 'Underline',
          iconCss: 'e-icons e-underline'
        }
      }
    ]
  }
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

When the orientation is set to **Row** a group may have a maximum of three collections each of which may contain any number of items. When the orientation is set to **Column** a group may have any number of collections, each of which may contain one large-sized item or three medium/small-sized items. If two large-sized items are specified, it automatically converts into two medium/small-sized items.

Defining group header

You can use the [header](#) property to set the name for each group header.

INDEX.JS

```

var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        collections: [{
            items: [{
                type: 'SplitButton',
                allowedSizes: ej.ribbon.RibbonItemSize.Large,
                splitButtonSettings: {
                    iconCss: 'e-icons e-paste',
                    items: [
                        { text: 'Keep Source Format' },
                        { text: 'Merge format' },
                        { text: 'Keep text only' },
                    ],
                },
                content: 'Paste',
            }
        ]
    }]
},
],

```

```

{
  header: 'Font',
  collections: [{
    items: [{
      type: 'Button',
      buttonSettings: {
        content: 'Bold',
        iconCss: 'e-icons e-bold',
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Italic',
        iconCss: 'e-icons e-italic',
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Underline',
        iconCss: 'e-icons e-underline',
      }
    }
  ]
}]
}];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Defining group icon

You can use the [groupIconCss](#) property to customize the icons in the group overflow button. When the ribbon size is adjusted, the group popup will appear.

INDEX.JS

```

var fontSize = ['8', '9', '10', '11', '12', '14', '16', '18', '20', '22',
'24', '26', '28', '36', '48', '72', '96'];
var fontStyle = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math',
'Courier New', 'Candara', 'Georgia', 'Impact', 'Segoe Print', 'Segoe
Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana', 'Windings'
];
var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        groupIconCss: 'e-icons e-paste',
        collections: [{
            items: [{
                type: 'SplitButton',
                splitButtonSettings: {
                    iconCss: 'e-icons e-paste',
                    items: [{ text: 'Keep Source Format' }, { text: 'Merge format' },
                    { text: 'Keep text only' } ]],
                content: 'Paste'
            }
        ]
    }, {
        items: [{
            type: 'Button',

```



```

        buttonSettings: {
            content: 'Cut',
            iconCss: 'e-icons e-cut'
        }
    }, {
        type: 'Button',
        buttonSettings: {
            content: 'Copy',
            iconCss: 'e-icons e-copy'
        }
    }, {
        type: 'Button',
        buttonSettings: {
            content: 'Format Painter',
            iconCss: 'e-icons e-format-painter'
        }
    }
    ]]
    ]]
    }, {
        header: 'Font',
        groupIconCss: 'e-icons e-bold',
        collections: [{
            items: [{
                type: 'ComboBox',
                comboBoxSettings: {
                    dataSource: fontStyle,
                    index: 3
                }
            }, {
                type: 'ComboBox',
                comboBoxSettings: {
                    dataSource: fontSize,
                    index: 3
                }
            }
        ]
    }, {
        items: [{
            type: 'Button',
            buttonSettings: {
                content: 'Bold',
                iconCss: 'e-icons e-bold'
            }
        }, {
            type: 'Button',
            buttonSettings: {
                content: 'Italic',
                iconCss: 'e-icons e-italic'
            }
        }, {
            type: 'Button',
            buttonSettings: {
                content: 'Underline',
                iconCss: 'e-icons e-underline'
            }
        }
    ]
    ]]
    ]]
    ]]

```

```

    }];
    var ribbon = new ej.ribbon.Ribbon({
        tabs: tabs
    });
    ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enabling group launcher icon

You can use the [showLauncherIcon](#) property to enable or disable the launcher icon for each group. By default, the property is set to `false`.

INDEX.JS

```
var tabs = [{
  header: "Home",
  groups: [{
    header: 'Clipboard',
    showLauncherIcon: true,
    collections: [{
      items: [{
        type: 'SplitButton',
        splitButtonSettings: {
          iconCss: 'e-icons e-paste',
          items: [{ text: 'Keep Source Format' }, { text: 'Merge
format' }, { text: 'Keep text only' }],
          content: 'Paste'
        }
      }
    ]
  }, {
    items: [{
      type: 'Button',
      buttonSettings: {
        content: 'Cut',
        iconCss: 'e-icons e-cut'
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Copy',
        iconCss: 'e-icons e-copy'
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Format Painter',
        iconCss: 'e-icons e-format-painter'
      }
    }
  ]
}]
}];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
```

```

<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize launcher icon

You can use the [launcherIconCss](#) property to customize the launcher icon by applying the custom styles.

INDEX.JS

```

var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        showLauncherIcon: true,

```

```

        collections: [{
            items: [{
                type: 'SplitButton',
                splitButtonSettings: {
                    iconCss: 'e-icons e-paste',
                    items: [{ text: 'Keep Source Format' }, { text: 'Merge
format' }, { text: 'Keep text only' }],
                    content: 'Paste'
                }
            }
        ]
    }, {
        items: [{
            type: 'Button',
            buttonSettings: {
                content: 'Cut',
                iconCss: 'e-icons e-cut'
            }
        }, {
            type: 'Button',
            buttonSettings: {
                content: 'Copy',
                iconCss: 'e-icons e-copy'
            }
        }, {
            type: 'Button',
            buttonSettings: {
                content: 'Format Painter',
                iconCss: 'e-icons e-format-painter'
            }
        }
    ]
}
    ]
    });
var ribbon = new ej.ribbon.Ribbon({
    launcherIconCss: 'e-icons e-description',
    tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Defining group collapsible state

You can use the [isCollapsible](#) property to determine whether the group is collapsed or not during resize. By default, the property is set to `true`. To prevent the group from being collapsed, set the property to `false`.

INDEX.JS

```

var fontSize = ['8', '9', '10', '11', '12', '14', '16', '18', '20', '22',
'24', '26', '28', '36', '48', '72', '96'];
var fontStyle = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math',
'Courier New', 'Candara', 'Georgia', 'Impact', 'Segoe Print', 'Segoe
Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana', 'Windings'
];
var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        groupIconCss: 'e-icons e-paste',
        collections: [{
            items: [{
                type: 'SplitButton',
                splitButtonSettings: {

```

```

        iconCss: 'e-icons e-paste',
        items: [{ text: 'Keep Source Format' }, { text: 'Merge format' },
{ text: 'Keep text only' }],
        content: 'Paste'
    }
    ]]
}, {
    items: [{
        type: 'Button',
        buttonSettings: {
            content: 'Cut',
            iconCss: 'e-icons e-cut'
        }
    }, {
        type: 'Button',
        buttonSettings: {
            content: 'Copy',
            iconCss: 'e-icons e-copy'
        }
    }, {
        type: 'Button',
        buttonSettings: {
            content: 'Format Painter',
            iconCss: 'e-icons e-format-painter'
        }
    }
    ]]
}}
}, {
    header: 'Font',
    isCollapsible: false,
    collections: [{
        items: [{
            type: 'ComboBox',
            comboBoxSettings: {
                dataSource: fontStyle,
                index: 3
            }
        }, {
            type: 'ComboBox',
            comboBoxSettings: {
                dataSource: fontSize,
                index: 3
            }
        }
    ]
}, {
    items: [{
        type: 'Button',
        buttonSettings: {
            content: 'Bold',
            iconCss: 'e-icons e-bold'
        }
    }, {
        type: 'Button',
        buttonSettings: {
            content: 'Italic',
            iconCss: 'e-icons e-italic'
        }
    }
}

```

```

    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Underline',
        iconCss: 'e-icons e-underline'
      }
    }
  ]
}
});
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ribbon"></div>

```



```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Defining priority order for group collapse or expand

You can use the [priority](#) property to set the priority order for each group which should be collapsed or expanded on resizing. When collapsing, higher priority values are fetched first. When expanding, lower priority values are fetched first.

INDEX.JS

```

var fontSize = ['8', '9', '10', '11', '12', '14', '16', '18', '20', '22',
'24', '26', '28', '36', '48', '72', '96'];
var fontStyle = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math',
'Courier New', 'Candara', 'Georgia', 'Impact', 'Segoe Print', 'Segoe
Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana', 'Windings'
];
var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        groupIconCss: 'e-icons e-paste',
        priority: 2,
        collections: [{
            items: [{
                type: 'SplitButton',
                splitButtonSettings: {
                    iconCss: 'e-icons e-paste',
                    items: [{ text: 'Keep Source Format' }, { text: 'Merge format' },
{ text: 'Keep text only' }],
                    content: 'Paste'
                }
            }
        ]
    }, {
        items: [{
            type: 'Button',
            buttonSettings: {
                content: 'Cut',
                iconCss: 'e-icons e-cut'
            }
        }, {
            type: 'Button',
            buttonSettings: {
                content: 'Copy',
                iconCss: 'e-icons e-copy'
            }
        }, {
            type: 'Button',
            buttonSettings: {
                content: 'Format Painter',

```

```

        iconCss: 'e-icons e-format-painter'
    }
}]]
}, {
header: 'Font',
groupIconCss: 'e-icons e-bold',
priority: 0,
collections: [{
items: [{
type: 'ComboBox',
comboBoxSettings: {
dataSource: fontStyle,
index: 3
}
}], {
type: 'ComboBox',
comboBoxSettings: {
dataSource: fontSize,
index: 3
}
}]]
}, {
items: [{
type: 'Button',
buttonSettings: {
content: 'Bold',
iconCss: 'e-icons e-bold'
}
}], {
type: 'Button',
buttonSettings: {
content: 'Italic',
iconCss: 'e-icons e-italic'
}
}], {
type: 'Button',
buttonSettings: {
content: 'Underline',
iconCss: 'e-icons e-underline'
}
}]]
}, {
header: 'Editing',
groupIconCss: 'e-icons e-edit',
priority: 1,
collections: [{
items: [{
type: 'SplitButton',
splitButtonSettings: {
iconCss: 'e-icons e-search',
items: [
{ text: 'Find', iconCss: 'e-icons e-search' },
{ text: 'Advanced find', iconCss: 'e-icons e-search' },
{ text: 'Go to', iconCss: 'e-icons e-arrow-right' }
]
}
}]]

```

```

        content: 'Find'
    }, {
        type: 'Button',
        buttonSettings: {
            content: 'Replace',
            iconCss: 'e-icons e-replace'
        }
    }, {
        type: 'SplitButton',
        splitButtonSettings: {
            iconCss: 'e-icons e-mouse-pointer',
            items: [
                { text: 'Select All' },
                { text: 'Select Objects' }
            ],
            content: 'Select'
        }
    }
]
}]]];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs
});
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
```

```

<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Simplified layout

In simplified layout, the Ribbon component organizes the items and groups into a single row by setting the [activeLayout](#) property to [Simplified](#).

INDEX.JS

```

var tabs = [
    {
        header: "Home",
        groups: [
            {
                header: "Clipboard",
                collections: [
                    {
                        items: [
                            {
                                type: "Button",
                                buttonSettings: {
                                    content: "Paste",
                                    iconCss: "e-icons e-paste",
                                }
                            },
                        ],
                    },
                ],
            },
            {
                items: [
                    {
                        type: "Button",
                        buttonSettings: {
                            content: "Cut",
                            iconCss: "e-icons e-cut",
                        }
                    },
                ],
            },
        ],
    },
];

```

```

        type: "Button",
        buttonSettings: {
            content: "Copy",
            iconCss: "e-icons e-copy",
        }
    },
],
}
]
}
],
},
{
    header: "Insert",
    groups: [
        {
            header: "Tables",
            collections: [{
                items: [{
                    type: "DropDown",
                    dropDownSettings: {
                        iconCss: "e-icons e-table",
                        content: "Table",
                        items: [
                            { text: "Insert Table" }, { text: "Draw Table" },
                            { text: "Convert Table" }, { text: "Excel
SpreadSheet" }
                        ]
                    }
                }
            ]
        }
    ]
}
];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    activeLayout: "Simplified"
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enabling group overflow popup

You can use the [enableGroupOverflow](#) property to add a separate popup for the overflow items in the group while resizing. The overflow items will appear in the common overflow popup, located at the right end of the tab content if it is set to **false**.

INDEX.JS

```

var fontSize = ['8', '9', '10', '11', '12', '14', '16', '18', '20', '22',
'24', '26', '28', '36', '48', '72', '96'];
var fontStyle = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math',
'Courier New', 'Candara', 'Georgia', 'Impact', 'Segoe Print', 'Segoe
Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana', 'Windings'
];
var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        groupIconCss: 'e-icons e-paste',

```

```

collections: [{
  items: [{
    type: 'SplitButton',
    splitButtonSettings: {
      iconCss: 'e-icons e-paste',
      items: [{ text: 'Keep Source Format' }, { text: 'Merge format' },
{ text: 'Keep text only' }],
      content: 'Paste'
    }
  ]
}, {
  items: [{
    type: 'Button',
    buttonSettings: {
      content: 'Cut',
      iconCss: 'e-icons e-cut'
    }
  }, {
    type: 'Button',
    buttonSettings: {
      content: 'Copy',
      iconCss: 'e-icons e-copy'
    }
  }, {
    type: 'Button',
    buttonSettings: {
      content: 'Format Painter',
      iconCss: 'e-icons e-format-painter'
    }
  ]
}]
}, {
  header: 'Font',
  groupIconCss: 'e-icons e-bold',
  orientation: 'Row',
  enableGroupOverflow: true,
  collections: [{
    items: [{
      type: 'ComboBox',
      comboBoxSettings: {
        dataSource: fontStyle,
        index: 3
      }
    }, {
      type: 'ComboBox',
      comboBoxSettings: {
        dataSource: fontSize,
        index: 3
      }
    }
  ]
}, {
  items: [{
    type: 'ColorPicker',
    allowedSizes: ej.ribbon.RibbonItemSize.Small,
    colorPickerSettings: {
      value: '#123456'
    }
  ]
}

```

```

    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Bold',
        iconCss: 'e-icons e-bold'
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Italic',
        iconCss: 'e-icons e-italic'
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Underline',
        iconCss: 'e-icons e-underline'
      }
    }
  ]
}
}, {
  header: 'Editing',
  groupIconCss: 'e-icons e-edit',
  collections: [{
    items: [{
      type: 'SplitButton',
      splitButtonSettings: {
        iconCss: 'e-icons e-search',
        items: [
          { text: 'Find', iconCss: 'e-icons e-search' },
          { text: 'Advanced find', iconCss: 'e-icons e-search' },
          { text: 'Go to', iconCss: 'e-icons e-arrow-right' }
        ],
        content: 'Find'
      }
    }, {
      type: 'Button',
      buttonSettings: {
        content: 'Replace',
        iconCss: 'e-icons e-replace'
      }
    }, {
      type: 'SplitButton',
      splitButtonSettings: {
        iconCss: 'e-icons e-mouse-pointer',
        items: [
          { text: 'Select All' },
          { text: 'Select Objects' }
        ],
        content: 'Select'
      }
    }
  ]
}
}
}];
var ribbon = new ej.ribbon.Ribbon({
  activeLayout: 'Simplified',

```



```

    tabs: tabs
  });
  ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Minimized State

You can hide the Ribbon contents and display only the tab headers by double-clicking on the tab header. In minimized state, the Ribbon control expands to its normal state when click on the tab header.

You can use the [isMinimized](#) property to change the Ribbon component to minimized state. By default, the value is `false`.

INDEX.JS

```
var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                }
              },
            ],
          },
        ],
      },
      {
        items: [
          {
            type: "Button",
            buttonSettings: {
              content: "Cut",
              iconCss: "e-icons e-cut",
            }
          },
          {
            type: "Button",
            buttonSettings: {
              content: "Copy",
              iconCss: "e-icons e-copy",
            }
          },
        ],
      },
    ],
  },
  {
    header: "Insert",
    groups: [
      {
        header: "Tables",
        collections: [{
          items: [{
            type: "DropDown",

```

```

        dropDownSettings: {
            iconCss: "e-icons e-table",
            content: "Table",
            items: [
                { text: "Insert Table" }, { text: "Draw Table"
SpreadSheet" }
                { text: "Convert Table" }, { text: "Excel
        ]
    }
    ]]
    ]]
    }
    ]
    }
};
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    isMinimized: true
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

File Menu

The Ribbon control provides a built-in file menu that allows you to add menu items for performing specific actions. The file menu can be enabled by setting the [fileMenu](#) property.

Visibility

You can show the file menu by setting the [visible](#) property to `true`. By default, the file menu is hidden.

INDEX.JS

```

var tabs = [
    {
        header: "Home",
        groups: [
            {
                header: "Clipboard",
                collections: [
                    {
                        items: [
                            {
                                type: "Button",
                                buttonSettings: {
                                    content: "Paste",
                                    iconCss: "e-icons e-paste",
                                }
                            }
                        ],
                    },
                ],
            },
            {
                items: [
                    {
                        type: "Button",
                        buttonSettings: {
                            content: "Cut",
                            iconCss: "e-icons e-cut",
                        }
                    },
                    {
                        type: "Button",
                        buttonSettings: {

```

```

        content: "Copy",
        iconCss: "e-icons e-copy",
    },
],
}
]
}
];

var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    fileMenu: {
        menuItems: [{ text: "New", iconCss: "e-icons e-file-new", id: "new"
    }],
        visible: true
    }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding menu items

The menu items can be added to the file menu using the [menuItems](#) property.

INDEX.JS

```

var tabs = [
{
    header: "Home",
    groups: [
        {
            header: "Clipboard",
            collections: [
                {
                    items: [
                        {
                            type: "Button",
                            buttonSettings: {
                                content: "Paste",
                                iconCss: "e-icons e-paste",
                            }
                        },
                    ],
                },
            ],
        },
        {
            items: [
                {
                    type: "Button",
                    buttonSettings: {
                        content: "Cut",
                        iconCss: "e-icons e-cut",
                    }
                },
                {
                    type: "Button",
                    buttonSettings: {
                        content: "Copy",
                        iconCss: "e-icons e-copy",
                    }
                },
            ],
        },
    ],
}

```

```

    ],
    }
  ]
}
]
}
];
var fileMenuItems = [
  { text: "New", iconCss: "e-icons e-file-new", id: "new" },
  { text: "Open", iconCss: "e-icons e-folder-open", id: "Open" },
  { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
  { text: "Save as", iconCss: "e-icons e-save", id: "save" }
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  fileMenu: {
    menuItems: fileMenuItems,
    visible: true
  }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open submenu on click

You can open the submenu on menu item click, by setting the [showItemOnClick](#) property to `true`. By default, the submenu will open on mouse hover.

INDEX.JS

```

var tabs = [
{
    header: "Home",
    groups: [
        {
            header: "Clipboard",
            collections: [
                {
                    items: [
                        {
                            type: "Button",
                            buttonSettings: {
                                content: "Paste",
                                iconCss: "e-icons e-paste",
                            },
                        },
                    ],
                },
            ],
        },
        {
            items: [
                {
                    type: "Button",
                    buttonSettings: {
                        content: "Cut",
                        iconCss: "e-icons e-cut",
                    },
                },
                {
                    type: "Button",
                    buttonSettings: {
                        content: "Copy",
                        iconCss: "e-icons e-copy",
                    },
                },
            ],
        },
    ],
},
]

```



```

    },
  ],
}
]
}
]
}
];
var fileMenuItems = [
  { text: "New", iconCss:"e-icons e-file-new", id: "new" },
  { text: "Open", iconCss:"e-icons e-folder-open", id: "open" },
  { text: "Rename", iconCss:"e-icons e-rename", id: "rename" },
  {
    text: "Save as",
    iconCss:"e-icons e-save",
    id: "save",
    items: [
      { text: "Microsoft Word (.docx)" },
      { text: "Microsoft Word 97-2003(.doc)" },
      { text: "Download as PDF" }
    ]
  }
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  fileMenu: {
    menuItems: fileMenuItems,
    showItemOnClick: true,
    visible: true
  }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom header text

You can define the file menu header text content by using the [text](#) property.

INDEX.JS

```

var tabs = [
    {
        header: "Home",
        groups: [
            {
                header: "Clipboard",
                collections: [
                    {
                        items: [
                            {
                                type: "Button",
                                buttonSettings: {
                                    content: "Paste",
                                    iconCss: "e-icons e-paste",
                                }
                            }
                        ],
                    },
                ],
            },
            {
                items: [
                    {
                        type: "Button",
                        buttonSettings: {

```

```

        content: "Cut",
        iconCss: "e-icons e-cut",
    }
},
{
    type: "Button",
    buttonSettings: {
        content: "Copy",
        iconCss: "e-icons e-copy",
    }
},
],
}
]
}
];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    fileMenu: {
        text: "App",
        menuItems: [{ text: "New", iconCss: "e-icons e-file-new", id: "new"
    }],
        visible: true
    }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Ribbon Backstage

The Ribbon supports backstage view which is an addition to the traditional file menu. It displays information like application settings, user details, etc. The backstage view can be enabled by setting the [backStageMenu](#) property.

The backstage view options are displayed on the left, while the content of each option is shown on the right.

Adding backstage items

The menu items can be added to the backstage view by using the [items](#) property. You can show the backstage view by setting the [visible](#) property to `true`. By default, the backstage view is hidden.

INDEX.JS

```

var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                },
              },
            ],
          },
        ],
      },
    ],
  },
],

```

```

        },
        {
            items: [
                {
                    type: "Button",
                    buttonSettings: {
                        content: "Cut",
                        iconCss: "e-icons e-cut",
                    }
                },
                {
                    type: "Button",
                    buttonSettings: {
                        content: "Copy",
                        iconCss: "e-icons e-copy",
                    }
                }
            ],
        }
    ],
}
]
}
];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    backstageMenu: {
        items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content: '#homeContent' },
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new', content: '#newContent' },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open', content: '#openContent' }
        ],
        visible: true,
        backButton: { text: 'Close' }
    }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script type="text/x-jsrender" id="homeContent">
    <div id='home-wrapper' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
        <div id='block-wrapper'><div class='section-title'> Recent </div>
        <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                        <td>
                            <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                            <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>

```

```

    </div>
</script>
<script type="text/x-jsrender" id="newContent">
    <div id='new-content' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
    </div>
</script>
<script type="text/x-jsrender" id="openContent">
    <div style="padding: 20px;">
        <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                        <td>
                            <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                            <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
        <div class="section-content" style="padding: 12px 0px cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                        <td>
                            <span style="display: block; font-size: 14px">
Protect Document </span>
                            <span style="font-size: 12px">To prevent
accidental changes,
                                this document has been set to open as view-
only.</span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

    }
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding footer items

You can use the [isFooter](#) property in the [items](#) collection to add the backstage view footer items. By default, the value is false.

INDEX.JS

```

var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                }
              },
            ],
          },
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Cut",
                  iconCss: "e-icons e-cut",
                }
              },
              {
                type: "Button",
                buttonSettings: {
                  content: "Copy",
                  iconCss: "e-icons e-copy",
                }
              },
            ],
          },
        ],
      },
    ],
  },
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  backstageMenu: {
    items: [

```



```

    { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
'#homeContent' },
    { id: 'new', text: 'New', iconCss: 'e-icons e-file-new', content:
'#newContent' },
    { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: '#openContent' },
    { separator: true, isFooter: true },
    { id: 'account', text: 'Account', isFooter: true, content:
'#accountContent' }
  ],
  visible: true,
  backButton: { text: 'Close' }
}
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="ribbon"></div>
    </div>
<script type="text/x-jsrender" id="homeContent">
  <div id='home-wrapper' style="padding: 20px;">
    <div id='new-section' class='new-wrapper'>
      <div class='section-title'> New </div>
      <div class='category_container'>
        <div class='doc_category_image'></div>
        <span class='doc_category_text'> New document </span>
      </div>
    </div>
    <div id='block-wrapper'><div class='section-title'> Recent </div>
    <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
      <table>
        <tbody>
          <tr>
            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
            <td>
              <span style="display: block; font-size:
14px"> Ribbon.docx </span>
              <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</script>
<script type="text/x-jsrender" id="newContent">
  <div id='new-content' style="padding: 20px;">
    <div id='new-section' class='new-wrapper'>
      <div class='section-title'> New </div>
      <div class='category_container'>
        <div class='doc_category_image'></div>
        <span class='doc_category_text'> New document </span>
      </div>
    </div>
  </div>
</script>
<script type="text/x-jsrender" id="openContent">
  <div style="padding: 20px;">
    <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
      <table>
        <tbody>
          <tr>
            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
            <td>
              <span style="display: block; font-size: 14px">
Open in Desktop App </span>

```

```

                                <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                                </td>
                                </tr>
                                </tbody>
                                </table>
                            </div>
                            <div class="section-content" style="padding: 12px 0px cursor:
pointer">
                                <table>
                                    <tbody>
                                        <tr>
                                            <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                                            <td>
                                                <span style="display: block; font-size: 14px">
Protect Document </span>
                                                <span style="font-size: 12px">To prevent
accidental changes,
                                                    this document has been set to open as view-
only.</span>
                                            </td>
                                        </tr>
                                    </tbody>
                                </table>
                            </div>
                        </div>
</script>
<script type="text/x-jsrender" id="accountContent">
    <div style="padding: 20px;">
        <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-people"></span>
</td>
                        <td>
                            <span style="display: block; font-size:
14px">Account type</span>
                            <span style="font-size:
12px">Administrator</span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
        <div class="section-content" style="padding: 12px 0px cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-
password"></span> </td>
                        <td>
                            <span style="display: block; font-size:
14px">Password protected</span>

```

```

        <span style="font-size: 12px">Yes</span>
      </td>
    </tr>
  </tbody>
</table>
</div>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding separator

The separators are horizontal lines used to separate the backstage view items. You can use the [separator](#) property to split the menu items.

INDEX.JS

```

var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                }
              },
            ],
          },
        ],
      },
      {
        items: [
          {
            type: "Button",
            buttonSettings: {
              content: "Cut",
              iconCss: "e-icons e-cut",
            }
          },
          {
            type: "Button",
            buttonSettings: {
              content: "Copy",
              iconCss: "e-icons e-copy",
            }
          }
        ]
      }
    ]
  }
]

```

```

    },
    ],
  }
]
}
]
}
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  backstageMenu: {
    items: [
      { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
'#homeContent' },
      { id: 'new', text: 'New', iconCss: 'e-icons e-file-new', content:
'#newContent' },
      { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: '#openContent' },
      { separator: true },
      { text: 'Print', content: '#printContent' }
    ],
    visible: true,
    backButton: { text: 'Close' }
  }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">

```

```

<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script type="text/x-jsrender" id="homeContent">
    <div id='home-wrapper' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
        <div id='block-wrapper'><div class='section-title'> Recent </div>
        <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                        <td>
                            <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                            <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</script>
<script type="text/x-jsrender" id="newContent">
    <div id='new-content' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
    </div>
</script>
<script type="text/x-jsrender" id="openContent">

```

```

<div style="padding: 20px;">
  <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
    <table>
      <tbody>
        <tr>
          <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
          <td>
            <span style="display: block; font-size: 14px">
Open in Desktop App </span>
            <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
  <div class="section-content" style="padding: 12px 0px cursor:
pointer">
    <table>
      <tbody>
        <tr>
          <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
          <td>
            <span style="display: block; font-size: 14px">
Protect Document </span>
            <span style="font-size: 12px">To prevent
accidental changes,
                                this document has been set to open as view-
only.</span>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
</script>
<script type="text/x-jsrender" id="printContent">
  <div style="min-width: 300px; padding: 20px;">
    <h2>Print this document</h2>
    <button id="togglebtn" class="e-control e-btn e-lib e-flat e-
primary">
      <span class="e-btn-icon e-btn-sb-icons e-icons e-print e-icon-left">
      </span>Print</button>
    </div>
  </script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Back button

You can use the [backButton](#) property to customize the text and icon of the close button using the [text](#) and [iconCss](#) property. You can show the back button by setting the [visible](#) property to `true`.

INDEX.JS

```
var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                }
              },
            ],
          },
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Cut",
                  iconCss: "e-icons e-cut",
                }
              },
              {
                type: "Button",
                buttonSettings: {
                  content: "Copy",
                  iconCss: "e-icons e-copy",
                }
              },
            ],
          },
        ],
      },
    ],
  }
];

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  visible: true,
  backStageMenu: {
    items: [
      { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
'#homeContent' }
    ],
  },
});
```



```

        backButton: {
            text: 'Close',
        }
    }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script type="text/x-jsrender" id="homeContent">
    <div id='home-wrapper' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>

```

```

        <div class='doc_category_image'></div>
        <span class='doc_category_text'> New document </span>
    </div>
</div>
<div id='block-wrapper'><div class='section-title'> Recent </div>
    <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
        <table>
            <tbody>
                <tr>
                    <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                    <td>
                        <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                        <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
</script>
<script type="text/x-jsrender" id="newContent">
    <div id='new-content' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
    </div>
</script>
<script type="text/x-jsrender" id="openContent">
    <div style="padding: 20px;">
        <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                        <td>
                            <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                            <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
        <div class="section-content" style="padding: 12px 0px cursor:
pointer">

```

```

        <table>
          <tbody>
            <tr>
              <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
              <td>
                <span style="display: block; font-size: 14px">
Protect Document </span>
                <span style="font-size: 12px">To prevent
accidental changes,
                    this document has been set to open as view-
only.</span>
              </td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Backstage target

The [target](#) property specifies the element selector in which backstage will be displayed. The target element should have the position as relative, else the backstage will be positioned nearest to the relative element. By default, the backstage is positioned to ribbon element.

INDEX.JS

```

var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                }
              },
            ],
          },
        ],
      },
    ],
  },
  {
    items: [
      {

```

```

        type: "Button",
        buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut",
        }
    },
    {
        type: "Button",
        buttonSettings: {
            content: "Copy",
            iconCss: "e-icons e-copy",
        }
    },
],
}
]
}
];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    backstageMenu: {
        target: '#targetElement',
        items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
'#homeContent' },
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new', content:
'#newContent' },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: '#openContent' }
        ],
        visible: true,
        backButton: { text: 'Close' }
    }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-ribbon/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
        <div id="targetElement"></div>
    </div>
<script type="text/x-jsrender" id="homeContent">
    <div id='home-wrapper' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
        <div id='block-wrapper'><div class='section-title'> Recent </div>
            <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                            <td>
                                <span style="display: block; font-size:
14px"> Ribbon.docx </span>
                                <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>

```

```

</script>
<script type="text/x-jsrender" id="newContent">
  <div id='new-content' style="padding: 20px;">
    <div id='new-section' class='new-wrapper'>
      <div class='section-title'> New </div>
      <div class='category_container'>
        <div class='doc_category_image'></div>
        <span class='doc_category_text'> New document </span>
      </div>
    </div>
  </div>
</script>
<script type="text/x-jsrender" id="openContent">
  <div style="padding: 20px;">
    <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
      <table>
        <tbody>
          <tr>
            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
            <td>
              <span style="display: block; font-size: 14px">
Open in Desktop App </span>
              <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
    <div class="section-content" style="padding: 12px 0px cursor:
pointer">
      <table>
        <tbody>
          <tr>
            <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
            <td>
              <span style="display: block; font-size: 14px">
Protect Document </span>
              <span style="font-size: 12px">To prevent
accidental changes,
                                this document has been set to open as view-
only.</span>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Template

You can use the [template](#) property to modify the backstage view menu items and their contents.

INDEX.JS

```
var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {
                type: "Button",
                buttonSettings: {
                  content: "Paste",
                  iconCss: "e-icons e-paste",
                }
              },
            ],
          },
        ],
      },
      {
        items: [
          {
            type: "Button",
            buttonSettings: {
              content: "Cut",
              iconCss: "e-icons e-cut",
            }
          },
          {
            type: "Button",
            buttonSettings: {
              content: "Copy",
              iconCss: "e-icons e-copy",
            }
          },
        ],
      },
    ],
  }
];

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  backstageMenu: {
    visible: true,
    template: '#templateContent'
  }
});
```

```
});
ribbon.appendTo("#ribbon");
var ribbonEle = document.getElementById('ribbon');
document.getElementById('ribbon_backstage').onclick = function () {
    if(ribbonEle) {
        ribbonEle.querySelector('#ribbon_backstagepopup').style.display =
        'block'
    }
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script type="text/x-jsrender" id="templateContent">
```



```

<div id="temp-content" style="width: 550px; height: 350px; display:
flex">
  <div id="items-wrapper" style="width: 130px; height:100%;
background: #779de8;">
    <ul>
      <li id="close" onclick="closeContent(this.id)">
        <span class="e-icons e-close"></span>
        Close
      </li>
      <li id="new" onclick="contentClick(this.id)">
        <span class="e-icons e-file-new"></span>
        New
      </li>
      <li id="open" onclick="contentClick(this.id)">
        <span class="e-icons e-folder-open"></span>
        Open
      </li>
      <li id="save" onclick="contentClick(this.id)">
        <span class="e-icons e-save"></span>
        Save
      </li>
    </ul>
  </div>
  <div id="content-wrapper">
    <div id='new-wrapper' class='content-open' style="padding:
20px;">
      <div id='new-section' class='new-wrapper'>
        <div class='section-title'> New </div>
        <div class='category_container'>
          <div class='doc_category_image'></div>
          <span class='doc_category_text'> New document
</span>
        </div>
      </div>
    </div>
    <div id="save-wrapper" class='content-close' style="padding:
20px;">
      <div class="section-content" style="padding: 12px 0px;
cursor: pointer">
        <table>
          <tbody>
            <tr>
              <td> <span class="doc_icon e-icons e-
save"></span> </td>
              <td>
                <span style="display: block; font-size:
14px"> Save as </span>
                <span style="font-size: 12px"> Save as
copy online </span>
              </td>
            </tr>
          </tbody>
        </table>
      </div>
      <div class="section-content" style="padding: 12px 0px
cursor: pointer">
        <table>

```

```
|  |  |
| --- | --- |
| </span> </td>  Rename </span>  Rename  this file. </span> | |

```

```

</div>
<div id="open-wrapper" class='content-close' style="padding: 20px;">

```

```

<div class="section-content" style="padding: 12px 0px; cursor: pointer">

```

```

|  |  |
| --- | --- |
| </span> </td>  Ribbon.docx </span>  EJ2 >>  Components >> Navigations >> Ribbon >> default </span> | |

```

```

</div>
<div class="section-content" style="padding: 12px 0px; cursor: pointer">

```

```

|  |  |
| --- | --- |
| </span> </td>  Classic_layout.docx </span>  EJ2 >>  Components >> Navigations >> Ribbon >> layouts </span> | |

```

```

</div>
<div class="section-content" style="padding: 12px 0px; cursor: pointer">

```

```

|
|  |

```

```

<td> <span class="doc_icon e-icons e-open-
link"></span> </td>
<td>
  <span style="display: block; font-size:
14px"> Simplified_layout.docx </span>
  <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> layouts </span>
</td>
</tr>
</tbody>
</table>
</div>
</div>
</div>
</div>
</div>
</script>
<script>
  function contentClick(id) {
    let ribbonEle = document.getElementById('ribbon');
    let content = ribbonEle.querySelector('.content-open')
    if(content) { content.classList.replace('content-open', 'content-
close'); }
    ribbonEle.querySelector('#' + id + '-
wrapper').classList.add('content-open');
  }
  function closeContent() {
    let ribbonEle = document.getElementById('ribbon');
    ribbonEle.querySelector('#ribbon_backstagepopup').style.display =
'none'
  }
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting width and height

You can customize the height and width of the backstage view using the [height](#) and [width](#) property. By default, dimensions are set based on the content added.

INDEX.JS

```

var tabs = [
  {
    header: "Home",
    groups: [
      {
        header: "Clipboard",
        collections: [
          {
            items: [
              {

```

```

        type: "Button",
        buttonSettings: {
            content: "Paste",
            iconCss: "e-icons e-paste",
        }
    },
],
},
{
    items: [
        {
            type: "Button",
            buttonSettings: {
                content: "Cut",
                iconCss: "e-icons e-cut",
            }
        },
        {
            type: "Button",
            buttonSettings: {
                content: "Copy",
                iconCss: "e-icons e-copy",
            }
        }
    ],
}
]
}
];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    backstageMenu: {
        items: [
            { id: 'home', text: 'Home', iconCss: 'e-icons e-home', content:
'#homeContent' },
            { id: 'new', text: 'New', iconCss: 'e-icons e-file-new', content:
'#newContent' },
            { id: 'open', text: 'Open', iconCss: 'e-icons e-folder-open',
content: '#openContent' }
        ],
        visible: true,
        backButton: { text: 'Close' }
    }
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en"><head>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js"></script>
<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script type="text/x-jsrender" id="homeContent">
    <div id='home-wrapper' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
        <div id='block-wrapper'><div class='section-title'> Recent </div>
            <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
                <table>
                    <tbody>
                        <tr>
                            <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                            <td>
                                <span style="display: block; font-size:
14px"> Ribbon.docx </span>

```

```

                                <span style="font-size: 12px"> EJ2 >>
Components >> Navigations >> Ribbon >> default </span>
                                </td>
                                </tr>
                                </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
</script>
<script type="text/x-jsrender" id="newContent">
    <div id='new-content' style="padding: 20px;">
        <div id='new-section' class='new-wrapper'>
            <div class='section-title'> New </div>
            <div class='category_container'>
                <div class='doc_category_image'></div>
                <span class='doc_category_text'> New document </span>
            </div>
        </div>
    </div>
</script>
<script type="text/x-jsrender" id="openContent">
    <div style="padding: 20px;">
        <div class="section-content" style="padding: 12px 0px; cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-open-
link"></span> </td>
                        <td>
                            <span style="display: block; font-size: 14px">
Open in Desktop App </span>
                            <span style="font-size: 12px"> Use the full
functionality of Ribbon </span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
        <div class="section-content" style="padding: 12px 0px cursor:
pointer">
            <table>
                <tbody>
                    <tr>
                        <td> <span class="doc_icon e-icons e-protect-
sheet"></span> </td>
                        <td>
                            <span style="display: block; font-size: 14px">
Protect Document </span>
                            <span style="font-size: 12px">To prevent
accidental changes,
                                this document has been set to open as view-
only.</span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</script>

```

```

        </table>
    </div>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[Adding Backstage events](#)

Help Pane

The help pane is dedicated area where the users can define help contents like controlling document permissions, sharing features, and more which appears on the right side of the Ribbon. You can use the [helpPaneTemplate](#) property to set the help pane contents.

INDEX.JS

```

var tabs = [
    {
        header: "Home",
        groups: [
            {
                collections: [
                    {
                        items: [
                            {
                                type: "Button",
                                buttonSettings: {
                                    content: "Cut",
                                    iconCss: "e-icons e-cut"
                                }
                            },
                        ],
                    },
                ],
            },
        ],
    },
];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    helpPaneTemplate: `<button class="action_btn"><label><span id="undo"
class="e-icons e-undo"></span> Undo </label></button>
<button class="action_btn"><label><span id="redo" class="e-icons e-
redo"></span> Redo </label></button>`
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">

  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="ribbon"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.action_btn {
  margin: 0px 3px;
  border: none;
}

```



```

    color: #ffffff;
    background-color: #0d6efd;
  }

  #undo, #redo{
    padding: 0px 3px ;
  }

```

Tooltip

The Ribbon component supports tooltip to show additional information in the Ribbon items. The tooltip appears when the user hovers over a Ribbon item.

Adding Title

You can use the [title](#) property to set the tooltip title for each Ribbon item.

INDEX.JS

```

var tabs = [{
  header: "Home",
  groups: [{
    header: 'Clipboard',
    collections: [{
      items: [{
        type: 'Button',
        ribbonTooltipSettings: {
          title: 'Cut'
        },
        buttonSettings: {
          content: 'Cut',
          iconCss: 'e-icons e-cut'
        }
      }]
    }, {
      items: [{
        type: 'Button',
        ribbonTooltipSettings: {
          title: 'Copy'
        },
        buttonSettings: {
          content: 'Copy',
          iconCss: 'e-icons e-copy'
        }
      }],
      {
        type: 'Button',
        ribbonTooltipSettings: {
          title: 'Paste'
        },
        buttonSettings: {
          content: 'Paste',
          iconCss: 'e-icons e-paste'
        }
      }
    }
  ]
}];
var ribbon = new ej.ribbon.Ribbon({

```

```

    tabs: tabs
  });
  ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding Content

You can use the [content](#) property to set the tooltip content for each Ribbon item.

INDEX.JS

```
var tabs = [{
  header: "Home",
  groups: [{
    header: 'Clipboard',
    collections: [{
      items: [{
        type: 'Button',
        ribbonTooltipSettings: {
          title: 'Cut',
          content: 'Cut selected text or objects'
        },
        buttonSettings: {
          content: 'Cut',
          iconCss: 'e-icons e-cut'
        }
      },
      {
        type: 'Button',
        ribbonTooltipSettings: {
          title: 'Copy',
          content: 'Copy selected text or objects'
        },
        buttonSettings: {
          content: 'Copy',
          iconCss: 'e-icons e-copy'
        }
      }
    ],
    {
      type: 'Button',
      ribbonTooltipSettings: {
        title: 'Paste',
        content: 'Paste copied or cut content'
      },
      buttonSettings: {
        content: 'Paste',
        iconCss: 'e-icons e-paste'
      }
    }
  ]
}]
};
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
  helper.js"></script>
```

```

<title>Essential JS 2 - Ribbon</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding Icon

You can use the [iconCss](#) property to specify the icons to be displayed in the tooltip.

INDEX.JS

```

var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        collections: [{

```

```

        items: [{
            type: 'Button',
            ribbonTooltipSettings: {
                title: 'Cut',
                content: 'Cut selected text or objects',
                iconCss: 'e-icons e-cut'
            },
            buttonSettings: {
                content: 'Cut',
                iconCss: 'e-icons e-cut'
            }
        }],
    }, {
        items: [{
            type: 'Button',
            ribbonTooltipSettings: {
                title: 'Copy',
                content: 'Copy selected text or objects',
                iconCss: 'e-icons e-copy'
            },
            buttonSettings: {
                content: 'Copy',
                iconCss: 'e-icons e-copy'
            }
        }],
    }, {
        type: 'Button',
        ribbonTooltipSettings: {
            title: 'Paste',
            content: 'Paste copied or cut content',
            iconCss: 'e-icons e-paste'
        },
        buttonSettings: {
            content: 'Paste',
            iconCss: 'e-icons e-paste'
        }
    }
    ]
    ]
    ]
    ];
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

You can use the [cssClass](#) property to customize the appearance of the tooltip with your own custom styles.

INDEX.JS

```

var tabs = [{
    header: "Home",
    groups: [{
        header: 'Clipboard',
        collections: [{
            items: [{
                type: 'SplitButton',
                splitButtonSettings: {
                    iconCss: 'e-icons e-paste',

```

```

        items: [{ text: 'Keep Source Format' }, { text: 'Merge
format' }, { text: 'Keep text only' }],
        content: 'Paste'
    },
    ribbonTooltipSettings: {
        title: 'Paste',
        content: 'Insert the clipboard content where the cursor
is currently placed.',
        cssClass: 'custom-tooltip'
    }
    ]
}, {
    items: [{
        type: 'Button',
        ribbonTooltipSettings: {
            title: 'Cut',
            content: 'Places the selected text or object on the
clipboard so that you can paste it somewhere else.',
            cssClass: 'custom-tooltip'
        },
        buttonSettings: {
            content: 'Cut',
            iconCss: 'e-icons e-cut'
        }
    }, {
        type: 'Button',
        ribbonTooltipSettings: {
            title: 'Copy',
            content: 'Copies the chosen text or object to the
clipboard so that you can reuse it elsewhere.',
            cssClass: 'custom-tooltip'
        },
        buttonSettings: {
            content: 'Copy',
            iconCss: 'e-icons e-copy'
        }
    }, {
        type: 'Button',
        ribbonTooltipSettings: {
            title: 'Format Painter',
            content: 'Copies the formatting style of a selected text
or object and applies it to other content within the document.',
            cssClass: 'custom-tooltip'
        },
        buttonSettings: {
            content: 'Format Painter',
            iconCss: 'e-icons e-format-painter'
        }
    }
    ]
}
    ]
    ]
};
var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs
});
ribbon.appendTo("#ribbon");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
    <title>Essential JS 2 - Ribbon</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
    <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}

```



```

}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
:root {
  --borderColor: rgb(72, 72, 72);
  --black: #000000;
}
/* To customize the appearance of the tooltip */
.custom-tooltip.e-ribbon-tooltip.e-popup {
  border: 2px solid var(--borderColor);
  border-radius: 5px;
  background: var(--black);
}
/* To customize the arrow of the tooltip */
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip .e-arrow-tip-inner.e-tip-top,
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip .e-arrow-tip-inner.e-tip-
bottom {
  color: var(--black);
}
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip-outer.e-tip-top {
  border-bottom: 8px solid var(--borderColor);
}
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip-outer.e-tip-bottom {
  border-top: 8px solid var(--borderColor);
}
/* To change the size of the tooltip title */
.custom-tooltip.e-ribbon-tooltip .e-tip-content .e-ribbon-tooltip-title {
  font-size: 14px;
}
/* To change the size of the tooltip content */
.custom-tooltip.e-ribbon-tooltip .e-tip-content .e-ribbon-text-container .e-
ribbon-tooltip-content {
  font-size: 11px;
}

```

Ribbon Resizing

The Ribbon effectively resizes the ribbon elements while being resized. It extends when the ribbon size is increased and collapses when the ribbon size is decreased. The resizing can be performed in both the classic and simplified modes. Also, we have an option to resize the ribbon elements in the custom order.

In classic mode on resizing, the items size will be changed based on the available width of the tab content from the order of Large-> Medium-> Small and viceversa.

In simplified mode on resizing, the items size will be changed based on the available width of the tab content from the order of Medium-> Small and viceversa.

Defining items allowed size

You can use the [allowedSizes](#) property to maintain a constant size for an item. If `allowedSizes` is set, it keeps the size constant even when being resized.

INDEX.JS

```
var tabs = [{
  header: "Home",
  groups: [{
    header: 'Clipboard',
    collections: [{
      items: [{
        type: 'Button',
        allowedSizes: ej.ribbon.RibbonItemSize.Large,
        buttonSettings: {
          content: 'Cut',
          iconCss: 'e-icons e-cut'
        }
      }]
    }]
  }]
}];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs
});
ribbon.appendTo("#ribbon");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-
helper.js"></script>
  <title>Essential JS 2 - Ribbon</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
ribbon/styles/material.css" rel="stylesheet">
  <!--system js reference and configuration-->
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="ribbon"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Defining items active size

You can use the [activeSize](#) property to define the item size initially, before it is being resized. When resized the [activeSize](#) property is updated based on the ribbon's overflow state, which is determined by the [allowedSizes](#) property being configured. By default, the value is [Medium](#).

Events

This section describes the ribbon events that will be triggered when appropriate actions are performed. The following events are available in the ribbon control.

tabSelected

The [tabSelected](#) event is triggered after selecting the tab item.

```
`js
```

```

var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }
      }
    ]
  }
}]

```

```

    }}
  }}
},
{
  header: "View",
  groups: [{
    header: "Views",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Print Layout",
            iconCss: "e-icons e-print"
          }
        }]
      }
    ]
  }];
  var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    tabSelected: (args) =>{
      // Your required action here
    }
  });
  ribbon.appendTo("#ribbon");
  ,

```

[tabSelecting](#)

The [tabSelecting](#) event is triggered before selecting the tab item.

```

`js
var tabs = [{
  header: "Home",
  groups: [{

```

```
header: "Clipboard",
collections: [
{
items: [{
type: "Button",
buttonSettings: {
content: "Cut",
iconCss: "e-icons e-cut"
}
}]
}],
{
header: "View",
groups: [{
header: "Views",
collections: [
{
items: [{
type: "Button",
buttonSettings: {
content: "Print Layout",
iconCss: "e-icons e-print"
}
}]
}]
}
];
var ribbon = new ej.ribbon.Ribbon({
tabs: tabs,
tabSelecting: (args) =>{
// Your required action here
```

```

}
});
ribbon.appendTo("#ribbon");
`

```

ribbonCollapsing

The [ribbonCollapsing](#) event is triggered before collapsing the ribbon.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }
      ]
    }
  ]
}];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  ribbonCollapsing: (args) =>{
    // Your required action here
  }
});
ribbon.appendTo("#ribbon");
`

```

ribbonExpanding

The [ribbonExpanding](#) event is triggered before expanding the ribbon.

```

`js

```

```

var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }]
      }
    ]
  }]
}];

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  ribbonExpanding: (args) =>{
    // Your required action here
  }
});

ribbon.appendTo("#ribbon");

```

launcherIconClick

The [launcherIconClick](#) event is triggered when the launcher icon of the group is clicked.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    showLauncherIcon: true,
    collections: [

```

```

{
  items: [{
    type: "Button",
    buttonSettings: {
      content: "Cut",
      iconCss: "e-icons e-cut"
    }
  }]
}]
});

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  launcherIconClick: (args) =>{
    // Your required action here
  }
});

ribbon.appendTo("#ribbon");

```

Button item events

clicked

The [clicked](#) event is triggered when the Button is clicked.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut",

```



```

clicked: () => {
// Your required action here
}
}
}}
}}
}}
}};
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

created

The [created](#) event is triggered when the Button is created.

```

`js
var tabs = [{
header: "Home",
groups: [{
header: "Clipboard",
collections: [
{
items: [{
type: "Button",
buttonSettings: {
content: "Cut",
iconCss: "e-icons e-cut",
created: () => {
// Your required action here
}
}
}
}
}
}
}];

```

```
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

CheckBox item events

change

The [change](#) event is triggered when the Checkbox state is changed.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "View",
    collections: [
      {
        items: [{
          type: "CheckBox",
          checkBoxSettings: {
            label: "Ruler",
            checked: false,
            change: () => {
              // Your required action here
            }
          }
        }
      ]
    ]
  }
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

created

The [created](#) event is triggered once the Checkbox is created.

```
`js
var tabs = [{
  header: "Home",
```

```

groups: [{
  header: "View",
  collections: [
    {
      items: [{
        type: "CheckBox",
        checkBoxSettings: {
          label: "Ruler",
          checked: false,
          created: () => {
            // Your required action here
          }
        }
      }]
    }
  ]
}];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

ColorPicker item events

change

The [change](#) event is triggered while changing the colors.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ColorPicker",
          colorPickerSettings: {
            value: "#fff",

```

```

change:(args) => {
// Your required action here
}
}
}}
}}
}}
}};
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

created

The [created](#) event is triggered once the ColorPicker is created.

```

`js
var tabs = [{
header: "Home",
groups: [{
header: "Font",
collections: [
{
items: [{
type: "ColorPicker",
colorPickerSettings: {
value: "#fff",
created:() => {
// Your required action here
}
}
}
}}
}}
}};
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });

```

```
ribbon.appendTo("#ribbon");
`
```

open

The [open](#) event is triggered while opening the ColorPicker popup.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ColorPicker",
          colorPickerSettings: {
            value: "#fff",
            open:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

select

The [select](#) event is triggered while selecting the color in picker/palette, when showButtons property is enabled.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
```

```
collections: [
{
items: [{
type: "ColorPicker",
colorPickerSettings: {
value: "#fff",
select:(args) => {
// Your required action here
}
}
}]
}
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

beforeClose

The [beforeClose](#) event is triggered before closing the ColorPicker popup.

```
`js
var tabs = [{
header: "Home",
groups: [{
header: "Font",
collections: [
{
items: [{
type: "ColorPicker",
colorPickerSettings: {
value: "#fff",
beforeClose:(args) => {
// Your required action here
}
}
}
}
}
}
}];
```

```

}
}}
}}
}}
});
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

beforeOpen

The [beforeOpen](#) event is triggered before opening the ColorPicker popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ColorPicker",
          colorPickerSettings: {
            value: "#123456",
            beforeOpen:(args) => {
              // Your required action here
            }
          }
        }
      }
    ]
  }
}
}
});
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

beforeTileRender

The [beforeTileRender](#) event is triggered while rendering each palette tile.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ColorPicker",
          colorPickerSettings: {
            value: "#123456",
            beforeTileRender:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

ComboBox item events

change

The [change](#) event is triggered when an item in a popup is selected or when the model value is changed by the user.

```

`js
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia"];

var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [

```



```

{
items: [{
type: "ComboBox",
comboBoxSettings: {
dataSource: fontStyle,
index: 3,
change:(args) => {
// Your required action here
}
}
}]
}
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

[close](#)

The [close](#) event is triggered when the popup is closed.

```

`js
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara",
"Georgia"];
var tabs = [{
header: "Home",
groups: [{
header: "Font",
collections: [
{
items: [{
type: "ComboBox",
comboBoxSettings: {
dataSource: fontStyle,
index: 3,
close:(args) => {

```

```
// Your required action here
}
}
}}
}}
}}
}};
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

open

The [open](#) event is triggered when the popup is opened.

```
`js
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara",
"Georgia"];
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ComboBox",
          comboBoxSettings: {
            dataSource: fontStyle,
            index: 3,
            open:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  }
}
}];
```

```
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

created

The [created](#) event is triggered once the Combobox is created.

```
`js
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara",
"Georgia"];
var tabs = [{
  header: "Home",
  groups: [{
    header: "Font",
    collections: [
      {
        items: [{
          type: "ComboBox",
          comboBoxSettings: {
            dataSource: fontStyle,
            index: 3,
            created:() => {
              // Your required action here
            }
          }
        }]
      }
    ]
  }]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

filtering

The [filtering](#) event triggers on typing a character in the Combobox.

```
`js
```

```

var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara",
"Georgia"];
var tabs = [{
header: "Home",
groups: [{
header: "Font",
collections: [
{
items: [{
type: "ComboBox",
comboBoxSettings: {
dataSource: fontStyle,
index: 3,
allowFiltering: true,
filtering: function (e) {
// Your required action here
}
}
}
}
}
}
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

select

The [select](#) event is triggered when an item in the popup is selected.

```
`js
```

```

var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara",
"Georgia"];
var tabs = [{
header: "Home",
groups: [{
header: "Font",

```

```
collections: [
{
items: [{
type: "ComboBox",
comboBoxSettings: {
dataSource: fontStyle,
index: 3,
select:(args) => {
// Your required action here
}
}
}]
}
}];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

beforeOpen

The [beforeOpen](#) event triggers before opening the popup.

```
`js
```

```
var fontStyle = ["Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara",
"Georgia"];

var tabs = [{
header: "Home",
groups: [{
header: "Font",
collections: [
{
items: [{
type: "ComboBox",
comboBoxSettings: {
dataSource: fontStyle,
index: 3,
```

```
beforeOpen:(args) => {  
  // Your required action here  
}  
  
}  
  
}}  
  
}}  
  
}}  
  
}};  
  
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });  
ribbon.appendTo("#ribbon");  
`
```

DropDown item events

beforeClose

The `beforeClose` event is triggered before closing the `DropDownButton` popup.

```
js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "DropDown",
          dropDownSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            beforeClose:(args) => {
              // Your required action here
            }
          }
        }
      }
    ]
  }
}]
}
```

```

});
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

beforeOpen

The [beforeOpen](#) event is triggered before opening the Dropdown button popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "DropDown",
          dropDownSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            beforeOpen:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

beforeItemRender

The [beforeItemRender](#) event is triggered while rendering each popup item of the Dropdown button.

```

`js
var tabs = [{

```

```

header: "Home",
groups: [{
  header: "Header & Footer",
  collections: [
    {
      items: [{
        type: "DropDown",
        dropDownSettings: {
          content: "Header",
          iconCss: "e-icons e-header",
          items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
          beforeItemRender:(args) => {
            // Your required action here
          }
        }
      }
    ]
  }
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

[open](#)

The [open](#) event is triggered while opening the Dropdown button popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "DropDown",

```



```
dropDownSettings: {
  content: "Header",
  iconCss: "e-icons e-header",
  items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
  open:(args) => {
    // Your required action here
  }
}
}}
}}
}};

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
\
```

close

The [close](#) event is triggered while closing the Dropdown button popup.

```
`js
```

```
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "DropDown",
          dropDownSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            close:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}
```

```

}
}}
}}
}}
});
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

created

The [created](#) event is triggered when the DropDown is created.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "DropDown",
          dropDownSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            created:() => {
              // Your required action here
            }
          }
        }
      }
    ]
  }
}
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

`

select

The [select](#) event is triggered while selecting an action item in the Dropdown button popup.

`js

```
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "DropDown",
          dropDownSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            select:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

SplitButton item events

beforeClose

The [beforeClose](#) event is triggered before closing the SplitButton popup.

`js

```
var tabs = [{
  header: "Home",
  groups: [{
```

```

header: "Header & Footer",
collections: [
{
items: [{
type: "SplitButton",
splitButtonSettings: {
content: "Header",
iconCss: "e-icons e-header",
items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
beforeClose:(args) => {
// Your required action here
}
}
}]
}
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

beforeOpen

The [beforeOpen](#) event is triggered before opening the SplitButton popup.

```

`js
var tabs = [{
header: "Home",
groups: [{
header: "Header & Footer",
collections: [
{
items: [{
type: "SplitButton",
splitButtonSettings: {
content: "Header",

```

```
iconCss: "e-icons e-header",
items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
beforeOpen:(args) => {
// Your required action here
}
}
}}
}}
}}
}};

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
\
```

beforeItemRender

The `beforeItemRender` event is triggered while rendering each popup item of SplitButton

```
js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "SplitButton",
          splitButtonSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            beforeItemRender:(args) => {
              // Your required action here
            }
          }
        }
      }
    ]
  }
}]
```

```

    }}
  }}
  });
  var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
  ribbon.appendTo("#ribbon");
  ,

```

open

The [open](#) event is triggered while opening the SplitButton popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "SplitButton",
          splitButtonSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            open:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}];
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
,

```

close

The [close](#) event is triggered while closing the SplitButton popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "SplitButton",
          splitButtonSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            close:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}];

var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");

```

created

The [created](#) event is triggered when the SplitButton is created.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {

```

```

items: [{
  type: "SplitButton",
  splitButtonSettings: {
    content: "Header",
    iconCss: "e-icons e-header",
    items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
    created:() => {
      // Your required action here
    }
  }
}]
});
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

select

The [select](#) event is triggered while selecting an action item in the SplitButton popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "SplitButton",
          splitButtonSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            select:(args) => {

```



```
// Your required action here
}
}
}}
}}
}}
}};
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`
```

click

The [click](#) event is triggered while clicking the primary button in the SplitButton.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Header & Footer",
    collections: [
      {
        items: [{
          type: "SplitButton",
          splitButtonSettings: {
            content: "Header",
            iconCss: "e-icons e-header",
            items: [{ text: "Insert Header" }, { text: "Edit Header" }, { text: "Remove Header" }],
            click:(args) => {
              // Your required action here
            }
          }
        }
      ]
    }
  ]
}];
```

```
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
,
```

GroupButton item events

beforeClick

The [beforeClick](#) event is triggered before selecting a button from the groupbutton items.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Paragraph",
    collections: [
      {
        items: [{
          type: "GroupButton",
          allowedSizes: ej.ribbon.RibbonItemSize.Small,
          groupButtonSettings: {
            selection: ej.ribbon.RibbonGroupButtonSelection.Multiple,
            items: [{
              iconCss: 'e-icons e-bold',
              content: 'Bold',
              beforeClick:(args) => {
                // Your required action here
              }
            },
            {
              iconCss: 'e-icons e-italic',
              content: 'Italic',
              selected: true,
              beforeClick:(args) => {
                // Your required action here
              }
            },
            {
```

```

iconCss: 'e-icons e-underline',
content: 'Underline',
beforeClick:(args) => {
// Your required action here
}
},
{
iconCss: 'e-icons e-strikethrough',
selected: true,
content: 'Strikethrough',
beforeClick:(args) => {
// Your required action here
}
}
}
}
}
}
}
}
};
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

[click](#)

The [click](#) event is triggered when selecting a button from the groupbutton items.

```

`js
var tabs = [{
header: "Home",
groups: [{
header: "Paragraph",
collections: [
{
items: [{
type: "GroupButton",

```

```
allowedSizes: ej.ribbon.RibbonItemSize.Small,
groupButtonSettings: {
  selection: ej.ribbon.RibbonGroupButtonSelection.Single,
  items: [{
    iconCss: 'e-icons e-align-left',
    content: 'Align Left',
    click:(args) => {
      // Your required action here
    }
  },
  {
    iconCss: 'e-icons e-align-center',
    content: 'Align Center',
    selected: true,
    click:(args) => {
      // Your required action here
    }
  },
  {
    iconCss: 'e-icons e-align-right',
    content: 'Align Right',
    click:(args) => {
      // Your required action here
    }
  },
  {
    iconCss: 'e-icons e-justify',
    selected: true,
    content: 'Justify',
    click:(args) => {
      // Your required action here
    }
  }
}]
```

```

}
}}
}}
}}
});
var ribbon = new ej.ribbon.Ribbon({ tabs: tabs });
ribbon.appendTo("#ribbon");
`

```

FileMenu events

beforeClose

The [beforeClose](#) event is triggered before closing the fileMenu popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }]
      }
    ]
  }]
}];
var menuItems = [
  { text: "New", iconCss: "e-icons e-file-new", id: "new" },
  { text: "Open", iconCss: "e-icons e-folder-open", id: "Open" },
  { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
  { text: "Save as", iconCss: "e-icons e-save", id: "save" }
];

```

```

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  fileMenu: {
    menuItems: menuItems,
    visible: true,
    beforeClose: (args) => {
      // Your required action here
    }
  }
});
ribbon.appendTo("#ribbon");

```

beforeOpen

The [beforeOpen](#) event is triggered before opening the fileMenu popup.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }]
      }
    ]
  }
}];

var menuItems = [
  { text: "New", iconCss: "e-icons e-file-new", id: "new" },

```

```

{ text: "Open", iconCss: "e-icons e-folder-open", id: "Open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  fileMenu: {
    menuItems: menuItems,
    visible: true,
    beforeOpen: (args) => {
      // Your required action here
    }
  }
});
ribbon.appendTo("#ribbon");
`

```

beforeItemRender

The [beforeItemRender](#) event is triggered while rendering each ribbon fileMenu item.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }]
      }
    ]
  }
}]
`

```

```

    }]
  });
  var menuItems = [
    { text: "New", iconCss: "e-icons e-file-new", id: "new" },
    { text: "Open", iconCss: "e-icons e-folder-open", id: "Open" },
    { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
    { text: "Save as", iconCss: "e-icons e-save", id: "save" }
  ];
  var ribbon = new ej.ribbon.Ribbon({
    tabs: tabs,
    fileMenu: {
      menuItems: menuItems,
      visible: true,
      beforeItemRender: (args) => {
        // Your required action here
      }
    }
  });
  ribbon.appendTo("#ribbon");
  `

```

open

The [open](#) event is triggered when the fileMenu popup is opened.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",

```



```

iconCss: "e-icons e-cut"
}
}}
}}
}};
var menuItems = [
{ text: "New", iconCss: "e-icons e-file-new", id: "new" },
{ text: "Open", iconCss: "e-icons e-folder-open", id: "Open" },
{ text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
{ text: "Save as", iconCss: "e-icons e-save", id: "save" }
];
var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  fileMenu: {
    menuItems: menuItems,
    visible: true,
    open: (args) => {
      // Your required action here
    }
  }
});
ribbon.appendTo("#ribbon");

```

close

The [close](#) event is triggered when the fileMenu popup is closed.

```

`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {

```

```

items: [{
  type: "Button",
  buttonSettings: {
    content: "Cut",
    iconCss: "e-icons e-cut"
  }
}]
});

var menuItems = [
  { text: "New", iconCss: "e-icons e-file-new", id: "new" },
  { text: "Open", iconCss: "e-icons e-folder-open", id: "Open" },
  { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
  { text: "Save as", iconCss: "e-icons e-save", id: "save" }
];

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  fileMenu: {
    menuItems: menuItems,
    visible: true,
    close: (args) => {
      // Your required action here
    }
  }
});

ribbon.appendTo("#ribbon");

```

select

The [select](#) event is triggered while selecting an item in the ribbon fileMenu.

```
`js
```

```

var tabs = [{
  header: "Home",

```

```

groups: [{
  header: "Clipboard",
  collections: [
    {
      items: [{
        type: "Button",
        buttonSettings: {
          content: "Cut",
          iconCss: "e-icons e-cut"
        }
      }]
    }
  ]
}];

var menuItems = [
  { text: "New", iconCss: "e-icons e-file-new", id: "new" },
  { text: "Open", iconCss: "e-icons e-folder-open", id: "Open" },
  { text: "Rename", iconCss: "e-icons e-rename", id: "rename" },
  { text: "Save as", iconCss: "e-icons e-save", id: "save" }
];

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  fileMenu: {
    menuItems: menuItems,
    visible: true,
    select: (args) => {
      // Your required action here
    }
  }
});

ribbon.appendTo("#ribbon");

```

Backstage view events

backStageItemClick

The [backStageItemClick](#) event is triggered when backstage item is selected.

```
`js
var tabs = [{
  header: "Home",
  groups: [{
    header: "Clipboard",
    collections: [
      {
        items: [{
          type: "Button",
          buttonSettings: {
            content: "Cut",
            iconCss: "e-icons e-cut"
          }
        }]
      }
    ]
  }]
}];

var ribbon = new ej.ribbon.Ribbon({
  tabs: tabs,
  backstageMenu: {
    items: [
      {
        id: 'home',
        text: 'Home',
        iconCss: 'e-icons e-home',
        content: '#homeContent',
        backStageItemClick: (args) => {
          // Your required action here
        }
      }
    ],
  },
});
```

```

visible: true,
backButton: { text: 'Close' }
}
});
ribbon.appendTo("#ribbon");
`

```

RichTextEditor

Getting started in EJ2 JavaScript Rich text editor control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization

The Essential JS 2 JavaScript components can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder `myapp` for Essential JS 2 JavaScript components.

Step 2: You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

Syntax:

Script: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js`

Styles: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css`

Example:

Script: `C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-richtexteditor/dist/global/ej2-richtexteditor.min.js`

Styles: `C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-richtexteditor/styles/material.css`

Step 3: Create a folder `myapp/resources` and copy/paste the global scripts and styles from the above installed location to `myapp/resources` location.

Step 4: Create a HTML page (index.html) in `myapp` location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Rich Text Editor</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Rich Text Editor's global script -->
<script src="resources/ej2-richtexteditor.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the `textarea` element and initiate the `Essential JS 2 Rich Text Editor` component in the `index.html` by using following code

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Rich Text Editor</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Rich Text Editor's global script -->
<script src="resources/ej2-richtexteditor.min.js" type="text/javascript"></script>
</head>
<body>
<!--Element which will render as RTE-->
<textarea id="defaultRTE">
</textarea>
<script>
// initialize Rich Text Editor component
var RichTextEditor = new ej.richtexteditor.RichTextEditor();
```

```
// Render initialized Rich Text Editor.
RichTextEditor.appendTo('#defaultRTE');
</script>
</body>
</html>
`
```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Rich Text Editor** component.

Using CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-richtexteditor/dist/global/ej2-richtexteditor.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-richtexteditor/styles/material.css>

Step 3: Create a HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `textarea` element and initiate the **Essential JS 2 Rich Text Editor** component in the `index.html` by using following code.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Rich Text Editor</title>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
```

```

<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Rich Text Editor's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/dist/global/ej2-richtexteditor.min.js"
type="text/javascript"></script>
</head>
<body>
<!--Element which will render as RTE-->
<textarea id="defaultRTE">
</textarea>
<script>
// initialize Rich Text Editor component
var RichTextEditor = new ej.richtexteditor.RichTextEditor();
// Render initialized Rich Text Editor.
RichTextEditor.appendTo('#defaultRTE')
</script>
</body>
</html>

```

Step 4: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Rich Text Editor** component.

Initialize from textarea

Initialize the Rich Text Editor on a textarea.

Now, add an HTML textarea element to act as the Rich Text Editor element in `index.html` using the following code.

```
[src/index.html]
```

```
`html
<!DOCTYPE html>
```



```

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Rich Text Editor's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/dist/global/ej2-richtexteditor.min.js"
type="text/javascript"></script>
</head>
<body>
<!--Element which will render as RTE-->
<textarea id="defaultRTE"></textarea>
</body>
</html>
`

```

Place the following Rich Text Editor code in the `index.js` file.

```

[src/index.js]
`javascript
// initialize Rich Text Editor component
var defaultRTE = new ej.richtexteditor.RichTextEditor({});
// render initialized Rich Text Editor

```

```
defaultRTE.appendTo("#defaultRTE");
```

```
,
```

Output will be displayed as follows

INDEX.JS

```
// initialize Rich Text Editor component
var defaultRTE = new ej.richtexteditor.RichTextEditor({});
// render initialized Rich Text Editor
defaultRTE.appendTo("#defaultRTE");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <textarea id="defaultRTE"></textarea>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

Initialize from ` ` element

Rich Text Editor can be initialized on div element as shown below

[src/index.html]

```
`html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Essential JS 2 Rich Text Editor</title>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Rich Text Editor's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/dist/global/ej2-richtexteditor.min.js"
type="text/javascript"></script>
</head>
<body>
<!--Element which will render as RTE-->
<div id="defaultRTE">
<p>The Rich Text Editor component is WYSIWYG ("what you see is what you get") editor that provides
the best user experience to create and update the content.Users can format their content using
standard toolbar commands.</p>
```

```

<p><b>Key features:</b></p>
<ul><li><p>Provides IFRAME and DIV modes</p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third-party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
<li><p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p></li>
</ul>
</div>
<script>
// initialize Rich Text Editor component
var RichTextEditor = new ej.richtexteditor.RichTextEditor({
height: '340px'
});
// Render initialized Rich Text Editor.
RichTextEditor.appendTo('#defaultRTE')
</script>
</body>
</html>
`

```

Initialize from `<IFRAME>` element

The Rich Text Editor's content is placed in an iframe and isolated from the rest of the page.

Initialize the Rich Text Editor on div element and set the `enable` field of `iframeSettings` property to true.

```
[src/index.html]
```

```

`html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Essential JS 2 Rich Text Editor</title>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>

```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Rich Text Editor's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-richtexteditor/dist/global/ej2-richtexteditor.min.js"
type="text/javascript"></script>
</head>
<body>
<!--Element which will render as RTE-->
<div id="defaultRTE">
<p>The Rich Text Editor component is WYSIWYG ("what you see is what you get") editor that provides
the best user experience to create and update the content.Users can format their content using
standard toolbar commands.</p>
<p><b>Key features:</b></p>
<ul><li><p>Provides IFRAME and DIV modes</p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
<li><p>Supports third-party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
<li><p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p></li>
</ul>
</div>
<script>
```

```
// initialize Rich Text Editor component
var RichTextEditor = new ej.richtexteditor.RichTextEditor({
  height: '340px',
  iframeSettings: {
    enable: true
  }
});
// Render initialized Rich Text Editor.
RichTextEditor.appendTo('#defaultRTE')
</script>
</body>
</html>
`
```

Module Injection

To create Rich Text Editor with additional features, inject the required modules. The following modules are used to extend Rich Text Editor's basic functionality.

- **Toolbar** - Inject this module to use Toolbar feature.
- **Link** - Inject this module to use link feature in Rich Text Editor.
- **Image** - Inject this module to use image feature in Rich Text Editor.
- **Table** - Inject this module to use table feature in Rich Text Editor.
- **Count** - Inject this module to use character count in Rich Text Editor.
- **HtmlEditor** - Inject this module to use Rich Text Editor as html editor.
- **MarkdownEditor** - Inject this module to use Rich Text Editor as markdown editor.
- **QuickToolbar** - Inject this module to use quick toolbar feature for the target element.
- **Resize** - Inject this module to use resize feature in Rich Text Editor.
- **FileManager** - Inject this module to use file browser feature in Rich Text Editor.
- **PasteCleanup** - Inject this module to use paste cleanup feature in Rich Text Editor.
- **FormatPainter** - Inject this module to use format painter feature in Rich Text Editor.
- **EmojiPicker** - Inject this module to use emoji picker feature in Rich Text Editor.

These modules should be injected into the Rich Text Editor using the **RichTextEditor.Inject** method.

Configure the Toolbar

Configure the toolbar with the tools using items field of the [toolbarSettings](#) property as your application requires.

INDEX.JS

```
// initialize Rich Text Editor component
var defaultRTE = new ej.richtexteditor.RichTextEditor({toolbarSettings: {
  items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
    'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
    'LowerCase', 'UpperCase', '|'],
```

```

        'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
        'Outdent', 'Indent', '|',
        'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
        'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
    });
    // render initialized Rich Text Editor
    defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
      <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
      <li><p>Provides a fully customizable toolbar.</p></li>

```

```

        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

| and **-** can insert a vertical and horizontal separator lines in the toolbar.

Insert images and links

The [image](#) module inserts an image into Rich Text Editor's content area, and the [link](#) module links external resources such as website URLs, to selected text in the Rich Text Editor's content, respectively.

The link inject module adds a link icon to the toolbar and the image inject module adds an image icon to the toolbar.

Specifies the items to be rendered in the quick toolbar based on the target element such image, link and text element. The quick toolbar opens to customize the element by clicking the target element.

INDEX.JS

```

// initialize Rich Text Editor component
var defaultRTE = new ej.richtexteditor.RichTextEditor({
    //Define the quicktoolbar items for image
    quickToolbarSettings: {
        image: [
            'Replace', 'Align', 'Caption', 'Remove', 'InsertLink',
            'OpenImageLink', '-',
            'EditImageLink', 'RemoveImageLink', 'Display', 'AltText',
            'Dimension'
        ],
    },
    //Define toolbar items
    toolbarSettings: {
        items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',

```



```

        'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
        'LowerCase', 'UpperCase', '|',
        'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
        'Outdent', 'Indent', '|',
        'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
        'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
    }
});
// render initialized Rich Text Editor
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>

```

```

        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Retrieve the formatted content

To retrieve the editor contents, use [value](#) property of Rich Text Editor.

```
`ts
```

```
var rteValue = defaultRTE.value;
```

```
,
```

Or, you can use the public method, [getHtml](#) to retrieve the Rich Text Editor content.

```
`ts
```

```
var rteValue = defaultRTE.getHtml();
```

```
,
```

To fetch the Rich Text Editor's text content, use [getText](#) method of Rich Text Editor.

```
`ts
```

```
var rteValue = defaultRTE.getText();
```

```
,
```

Retrieve the number of characters in the Rich Text Editor

To get the maximum number of characters in the Rich Text Editor's content, use [getCharCount](#)

```
`ts
```

```
let rteCount: number = defaultRTE.getCharCount();
```

You can refer to our [JavaScript Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Rich Text Editor example](#) that shows how to render the rich text editor tools.

See Also

- [How to change the editor type](#)
- [How to render the iframe](#)
- [How to render the toolbar in inline mode](#)
- [How to insert Emoticons](#)
- [Blog posting using Rich Text Editor](#)
- [Reactive Form with Rich Text Editor](#)

Editor modes in EJ2 JavaScript Rich text editor control

The Rich Text Editor component used to create and edit the content and return valid HTML markup or markdown (MD) of the content. It supports the following two editing formation.

- HTML Editor
- Markdown Editor

HTML editor

Rich Text Editor is a WYSIWYG editing control for formatting the word content as HTML.

The HTML editing mode is the default mode in Rich Text Editor to format the content through the available toolbar items to return the valid HTML markup. Set the [editorMode](#) property as **HTML**.

To create Rich Text Editor with HTML editing feature, inject the [htmleditor](#) module to the Rich Text Editor using the `RichTextEditor.Inject(HtmlEditor)` method.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor } from
 '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor);
// initialize RichTextEditor component
let defaultRTE: RichTextEditor = new RichTextEditor({
  value: `

The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
  <p><b>Key features:</b></p>
  <ul><li><p>Provides <b>IFRAME</b> and <b>DIV</b> modes</p></li>
  <li><p>Capable of handling markdown editing.</p></li>
  <li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
  <li><p>Provides a fully customizable toolbar.</p></li>
  <li><p>Supports third-party library integration.</p></li>
  </ul>`


```

```
});
// render initialized Rich Text Editor
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
      <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
      <li><p>Provides a fully customizable toolbar.</p></li>
      <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
      <li><p>Supports third-party library integration.</p></li>
```

```

        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Markdown editor

Set the [editorMode](#) property as **Markdown** to create or edit the content and apply formatting to view markdown formatted content.

The third-party library such as **Marked** or any other library is used to convert markdown into HTML content.

- The Supported Tags are **h6,h5,h4,h3,h2,h1,blockquote,pre,p,OL,UL**.
- The Supported Selection Tags are **Bold, Italic, StrikeThrough, InlineCode, SubScript, SuperScript, UpperCase, LowerCase**.

INDEX.TS

```

import { RichTextEditor, Toolbar, MarkdownEditor } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, MarkdownEditor);
let defaultRTE: RichTextEditor = new RichTextEditor({
    value: `***Overview***
    The Rich Text Editor component is WYSIWYG ("what you see is what you get")
    editor used to create and edit the content and return valid HTML markup or
    markdown (MD) of the content. The editor provides a standard toolbar to
    format content using its commands. Modular library features to load the
    necessary functionality on demand. The toolbar contains commands to align
    the text, insert link, insert image, insert list, undo/redo operation, HTML
    view, and more.
    ***Key features***
    - *Mode*: Provides IFRAME and DIV mode.
    - *Module*: Modular library to load the necessary functionality on demand.
    - *Toolbar*: Provide a fully customizable toolbar.

```

```

- *Editing*: HTML view to edit the source directly for developers.
- *Third-party Integration*: Supports to integrate third-party library.
- *Preview*: Preview the modified content before saving it.
- *Tools*: Handling images, hyperlinks, video, uploads and more.
- *Undo and Redo*: Undo/redo manager.
- *Lists*:Creates bulleted and numbered list.`
    editorMode: 'Markdown',
  });
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>

```

```

        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To create Rich Text Editor with Markdown editing feature, inject the [MarkdownEditor](#) module to the Rich Text Editor using the `RichTextEditor.Inject(MarkdownEditor)` method.

For further details on Markdown editing, refer to the [Markdown](#) section.

See Also

- [How to integrate the third party library](#)
- [How to render the iframe](#)

Toolbar in EJ2 JavaScript Rich text editor control

The Rich Text Editor toolbar contains a collection of tools such as bold, italic, and text alignment buttons that are used to format the content. However, in most integrations, you can customize the toolbar configurations easily to suit your needs.

To create Rich Text Editor with Markdown editing feature, inject the [Toolbar](#) module to the Rich Text Editor using the `RichTextEditor.Inject(Toolbar)` method.

The Rich Text Editor allows you to configure different types of toolbar using [toolbarSettings.type](#) property. The types of toolbar are:

1. Expand

2. MultiRow

Expand Toolbar

The default mode of [toolbarSettings.type](#) it will hide the overflowing items in the next row. By clicking the expand arrow, view the overflowing toolbar items.

INDEX.TS

```
import { RichTextEditor, Toolbar, HtmlEditor } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, HtmlEditor);
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    value: `<p>The Rich Text Editor component is WYSIWYG ("what you see is what you get") editor that provides the best user experience to create and update the content.Users can format their content using standard toolbar commands.</p>
    <p><b>Key features:</b></p>
    <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
    <li><p>Capable of handling markdown editing.</p></li>
    <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
    <li><p>Provides a fully customizable toolbar.</p></li>
    <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
    <li><p>Supports third-party library integration.</p></li>
    <li><p>Allows preview of modified content before saving
it.</p></li>
    <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>`,
    toolbarSettings: {
        type: 'Expand'
    });
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multi-row Toolbar

Set the type as MultiRow in [toolbarSettings](#) to hide the overflowing items in the next row. All toolbar items are visible.

INDEX.TS

```

import { RichTextEditor, Toolbar, HtmlEditor } from '@syncfusion/ej2-
richtexteditor';
RichTextEditor.Inject(Toolbar, HtmlEditor);
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    value: `<p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
    <p><b>Key features:</b></p>
    <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
    <li><p>Capable of handling markdown editing.</p></li>
    <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
    <li><p>Provides a fully customizable toolbar.</p></li>
    <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
    <li><p>Supports third-party library integration.</p></li>
    <li><p>Allows preview of modified content before saving it.</p></li>
    <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>

```

```

        <li><p>Contains undo/redo manager.</p></li>
        <li><p>Creates bulleted and numbered lists.</p></li>
    </ul>`,
    toolbarSettings: {
        type: 'MultiRow'
    });
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Floating Toolbar

By default, toolbar is float at the top of the Rich Text Editor on scrolling. It can be customized by specifying the offset of the floating toolbar from documents top position using [floatingToolbarOffset](#).

Enable or disable the floating toolbar using [enableFloating](#) of the [toolbarSettings](#) property.

INDEX.TS

```
import { RichTextEditor, Toolbar, HtmlEditor } from '@syncfusion/ej2-richtexteditor';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
RichTextEditor.Inject(Toolbar, HtmlEditor);
let defaultRTE: RichTextEditor = new RichTextEditor({
  height: 340,
  toolbarSettings: {
    enableFloating: false
  },
  value: `<p>The Rich Text Editor component is WYSIWYG ("what you see is what you get") editor that provides the best user experience to create and update the content.Users can format their content using standard toolbar commands.</p>
  <p><b>Key features:</b></p>
  <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62; modes</p></li>
  <li><p>Capable of handling markdown editing.</p></li>
  <li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
  <li><p>Provides a fully customizable toolbar.</p></li>
  <li><p>Provides HTML view to edit the source directly for developers.</p></li>
  <li><p>Supports third-party library integration.</p></li>
  <li><p>Allows preview of modified content before saving it.</p></li>
  <li><p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p></li>
  </ul>`;
});
defaultRTE.appendTo('#defaultRTE');
let float: CheckBox = new CheckBox({
  // set false for enable the checked state at initial rendering
  checked: false,
  label: 'Enable Floating',
  // bind change event
  change: (args: ChangeEventArgs) => {
    defaultRTE.toolbarSettings.enableFloating = (args as any).checked;
    defaultRTE.dataBind();
  }
});
float.appendTo('#float');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">




<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>












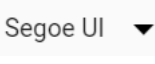
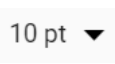


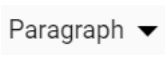

    <div id="container">
        <div id="defaultRTE">
            <input type="checkbox" id="float" checked="false">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>


```


Toolbar items


The following table lists the tools available in the toolbar.


Name	Icons	Summary	Initialization
Undo		Allows to undo the actions.	toolbarSettings: { items: ['Undo']}
Redo		Allows to redo the actions.	toolbarSettings: { items: ['Redo']}
Alignment		Align the content with left, center, and right margin.	toolbarSettings: { items: ['Alignments']}


	OrderedList		Create a new list item(numbered). toolbarSettings: { items: ['OrderedList']}
	UnorderedList		Create a new list item(bulleted). toolbarSettings: { items: ['UnorderedList']}
	Indent		Allows to increase the indent level of the content. toolbarSettings: { items: ['Indent']}
	Outdent		Allows to decrease the indent level of the content. toolbarSettings: { items: ['Outdent']}
	Hyperlink		Creates a hyperlink to a text or image to a specific location in the content. toolbarSettings: { items: ['CreateLink']}
	Images		Inserts an image from an online source or local computer. toolbarSettings: { items: ['Image']}
	LowerCase		Change the case of selected text to lower in the content. toolbarSettings: { items: ['LowerCase']}
	UpperCase		Change the case of selected text to upper in the content. toolbarSettings: { items: ['UpperCase']}
	SubScript		Makes the selected text as subscript (lower). toolbarSettings: { items: ['SubScript']}
	SuperScript		Makes the selected text as superscript (higher). toolbarSettings: { items: ['SuperScript']}
	Print		Allows to print the editor content. toolbarSettings: { items: ['Print']}
	FontName		Defines the fonts that appear under the Font Family DropDownList from the Rich Text Editor's toolbar. toolbarSettings: { items: ['FontName']}
	FontSize		Defines the font sizes that appear under the Font Size DropDownList from the Rich Text Editor's toolbar. toolbarSettings: { items: ['FontSize']}
	FontColor		Specifies an array of colors can be used in the colors popup for font color. toolbarSettings: { items: ['FontColor']}
	BackgroundColor		Specifies an array of colors can be used in the colors popup for background color. toolbarSettings: { items: ['BackgroundColor']}
	Format		An Object with the options that will appear in the Paragraph Format dropdown from the toolbar. toolbarSettings: { items: ['Formats']}
	StrikeThrough		Apply double line strike through formatting for the selected text. toolbarSettings: { items: ['StrikeThrough']}


| ClearFormat |  | The clear format tool is useful to remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. As a result, all the text formatting will be cleared and return to its default formatting styles. | toolbarSettings: { items: ['ClearFormat']} |


| FullScreen |  | Stretches the editor to the maximum width and height of the browser window. | toolbarSettings: { items: ['FullScreen']} |


| SourceCode |  | Rich Text Editor includes the ability for users to directly edit HTML code via “Source View”. If you made any modification in Source view directly, synchronize with Design view. | toolbarSettings: { items: ['SourceCode']} |


| NumberFormatList |  | Allows to create list items with various list style types(numbered). | toolbarSettings: { items: ['NumberFormatList']} |


| BulletFormatList |  | Allows to create list items with various list style types(bulleted). | toolbarSettings: { items: ['BulletFormatList']} |


| JustifyLeft |  | Allows each line to begin at the same distance from the editor’s left-hand side. | toolbarSettings: { items: ['JustifyLeft']} |


| JustifyCenter |  | There is an even space on each side of each line since the text is not aligned to the left or right margins. | toolbarSettings: { items: ['JustifyCenter']} |


| JustifyRight |  | Allows each line to end at the same distance from the editor’s right-hand side. | toolbarSettings: { items: ['JustifyRight']} |


| JustifyFull |  | The text is aligned with both right and left margins. | toolbarSettings: { items: ['JustifyFull']} |


| Bold |  | Text that is thicker and darker than usual. | toolbarSettings: { items: ['Bold']} |


| Italic |  | Shows a text that is leaned to the right. | toolbarSettings: { items: ['Italic']} |


| Underline |  | The underline is added to the selected text. | toolbarSettings: { items: ['Underline']} |


| ClearAll |  | Removes all styles that have been applied to the selected text. | toolbarSettings: { items: ['ClearAll']} |


| Cut |  | Removes the text from its current location and places it into the clipboard. | toolbarSettings: { items: ['Cut']} |


| Copy |  | The selected item is copied and pasted into the clipboard. | toolbarSettings: { items: ['Copy']} |


| Paste |  | Allows you to insert a clipboard item into a specific location. | toolbarSettings: { items: ['Paste']} |


| OpenLink |  | To open the URL link that is attached to the selected text. | toolbarSettings: { items: ['OpenLink']} |


| EditLink |  | Allows you to change the URL that has been attached to a specific item. | toolbarSettings: { items: ['EditLink']} |


| CreateTable |  | Create a table with defined columns and rows. | toolbarSettings: { items: ['CreateTable']} |


| RemoveTable |  | Removes the selected table and its contents. | toolbarSettings: { items: ['TableRemove']} |


| Replace |  | Replace the selected image with another image. | toolbarSettings: { items: ['Replace']} |


| Align |  | The image can be aligned to the right, left, or center. | toolbarSettings: { items: ['Align']} |


| Remove |  | Allows to remove the selected image from the editor. | toolbarSettings: { items: ['Remove']} |


| OpenImageLink |  | Opens the link that is attached to the selected image. | toolbarSettings: { items: ['OpenImageLink']} |


| EditImageLink |  | Allows to edit the link that is attached to the selected image. | toolbarSettings: { items: ['EditImageLink']} |


| RemoveImageLink |  | Removes the link that is attached to the selected image. | toolbarSettings: { items: ['RemoveImageLink']} |


| InsertLink |  | Allows users to add a link to a particular item. | toolbarSettings: { items: ['InsertLink']} |


| Display |  | Allows you to choose whether an image should be shown inline or as a block. | toolbarSettings: { items: ['Display']} |


| AltText |  | To display image description when an image on a Web page cannot be displayed. | toolbarSettings: { items: ['AltText']} |


| Dimension |  | Allows you to customize the image's height and width. | toolbarSettings: { items: ['Dimension']} |


| Maximize |  | Stretches the editor to the maximum width and height of the browser window. | toolbarSettings: { items: ['Maximize']} |


| Minimize |  | Shrinks the editor to the default width and height. | toolbarSettings: { items: ['Minimize']} |


| Preview |  | Allows to see how the editor's content looks in a browser. | toolbarSettings: { items: ['Preview']} |


| InsertCode |  | Represents preformatted text which is to be presented exactly as written in the HTML file. | toolbarSettings: { items: ['InsertCode']} |

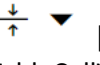
| RemoveLink |  | Allows you to remove the applied link from the selected item. | toolbarSettings: { items: ['RemoveLink']} |


| TableHeader |  | Allows you to add a table header. | toolbarSettings: { items: ['TableHeader']} |

| TableColumns |  | Shows the dropdown to insert a column or delete the selected column. | toolbarSettings: { items: ['TableColumns']} |

| TableRows |  | Shows the dropdown to insert a row or delete the selected row. | toolbarSettings: { items: ['TableRows']} |

| TableCellHorizontalAlign |  | Allows the table cell content to be aligned horizontally. | toolbarSettings: { items: ['TableCellHorizontalAlign']} |

| TableCellVerticalAlign |  | Allows the table cell content to be aligned vertically. | toolbarSettings: { items: ['TableCellVerticalAlign']} |

| TableEditProperties |  | Allows you to change the table width, padding, and cell spacing styles. | toolbarSettings: { items: ['TableEditProperties']} |

Note: The Paste toolbar button action will not work and the clipboard will be restricted due to some limitations in the browser. Pasting copied contents into the Rich Text Editor can be done by using the Ctrl+V keyboard command.

By default, tools will be arranged in following order.

```
items: ['Bold', 'Italic', 'Underline', '|', 'Formats', 'Alignments', 'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', '|', 'SourceCode', 'Undo', 'Redo']
```

The tools order can be customized as per application requirement. If you are not specifying any tools order, the editor will create the toolbar with default items.

Custom tool

The Rich Text Editor allows you to configure your own commands to its toolbar using the [toolbarSettings](#) property. The command can be plain text, icon, or HTML template. The order and the group can also be defined where the command should be included. Bind the action to the command by getting its instance.

This sample shows how to add your own commands to the toolbar of the Rich Text Editor. The "Ω" command is added to insert special characters in the editor. By clicking the "Ω" command, it will show the special characters list, and then choose the character to be inserted in the editor.

The following code snippet illustrates custom tool with tooltip text which will be included in [items](#) field of the [toolbarSettings](#) property.

In the following sample, once Rich Text Editor control is created, the concern event will be [created](#) the Dialog component can be rendered and target as RTE content.

```
`javascript
{
  tooltipText: 'Insert Symbol',
  undo: true,
  click: function() {
  },
  template: '<button class="e-tbar-btn e-btn" tabindex="-1" id="custom_tbar" style="width:100%"><div
class="e-tbar-btn-text" style="font-weight: 500;"> &#937;</div></button>'
}
`
```

Click the Ω command to show the special characters list, and then choose the character to be inserted in the editor.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Toolbar, Link, NodeSelection, Image, QuickToolbar,
HtmlEditor } from '@syncfusion/ej2-richtexteditor';
import { Dialog } from '@syncfusion/ej2-popups';
RichTextEditor.Inject(Toolbar, Link, Image, QuickToolbar, HtmlEditor);
let dialog: Dialog;
let defaultRTE: RichTextEditor = new RichTextEditor({
  height: 340,
  value: `<p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
<p><b>Key features:</b></p>
<ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62; modes</p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary functionality on
demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
```

```

<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third-party library integration.</p></li>
<li><p>Allows preview of modified content before saving it.</p></li>
<li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
</ul>`,
  toolbarSettings: {
    items: ['Bold', 'Italic', 'Underline', '|', 'Formats', 'Alignments',
'OrderedList',
    'UnorderedList', '|', 'CreateLink', 'Image', '|', 'SourceCode',
    {
      tooltipText: 'Insert Symbol',
      undo: true,
      click: function() {
        defaultRTE.focusIn();
        dialog.element.style.display = '';
        ranges = selection.getRange(document);
        dialog.width = defaultRTE.element.offsetWidth * 0.5;
        dialog.dataBind();
        dialog.show();
      },
      template: '<button class="e-tbar-btn e-btn" tabindex="-1"
id="custom_tbar" style="width:100%"><div class="e-tbar-btn-text"
style="font-weight: 500;"> &#937</div></button>'
    }, '|', 'Undo', 'Redo'
  ]
},
  created: onCreate
});
defaultRTE.appendTo('#defaultRTE');
let selection: NodeSelection = new NodeSelection();
let ranges: Range;
function onCreate(): void {
  let customBtn: HTMLElement =
defaultRTE.element.querySelector('#custom_tbar') as HTMLElement;
  let dialogCtn: HTMLElement = document.getElementById('rteSpecial_char');
  // Initialization of Dialog
  dialog = new Dialog({
    header: 'Special Characters',
    content: dialogCtn,
    target: document.getElementById('container'),
    showCloseIcon: false,
    isModal: true,
    height: 'auto',
    width: '500px',
    cssClass: 'e-rte-elements',
    overlayClick: dialogOverlay,
    buttons: [{ buttonModel: { content: "Insert", isPrimary: true },
click: onInsert }, { buttonModel: { content: 'Cancel' }, click:
dialogOverlay }]
  });
  // Render initialized Dialog
  dialog.appendTo('#customTbarDialog');
  dialog.hide();
  customBtn.onclick = (e: Event) => {
    dialog.element.style.display = '';

```

```

    ranges = selection.getRange(document);
    dialog.width = defaultRTE.element.offsetWidth * 0.5;
    dialog.dataBind();
    dialog.show();
    let dialogCtn: HTMLElement = document.getElementById('rteSpecial_char');
    dialogCtn.onclick = (e: Event) => {
        let target: HTMLElement = e.target as HTMLElement;
        let activeEle: HTMLElement =
    dialog.element.querySelector('.char_block.e-active');
        if (target.classList.contains('char_block')) {
            target.classList.add('e-active');
            if (activeEle) {
                activeEle.classList.remove('e-active');
            }
        }
    };
}
function onInsert(): void {
    let activeEle: HTMLElement =
    dialog.element.querySelector('.char_block.e-active');
    if (activeEle) {
        defaultRTE.executeCommand('insertText', activeEle.textContent,
{undo: true});
    }
    dialogOverlay();
}
function dialogOverlay(): void {
    let activeEle: HTMLElement =
    dialog.element.querySelector('.char_block.e-active');
    if (activeEle) {
        activeEle.classList.remove('e-active');
    }
    dialog.hide();
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="e-rte-custom-tbar-section">
    <div id="defaultRTE">
    </div>
    <div id="customTbarDialog" style="display: none">
      <div id="rteSpecial_char">
        <div class="char_block" title="^">^</div>
        <div class="char_block" title="_">_</div>
        <div class="char_block" title="`">`</div>
        <div class="char_block" title="{">{</div>
        <div class="char_block" title="|">|</div>
        <div class="char_block" title="}">}</div>
        <div class="char_block" title="~">~</div>
        <div class="char_block" title="&#160;">&#160;</div>
        <div class="char_block" title=";">;</div>
        <div class="char_block" title="ç">ç</div>
        <div class="char_block" title="£">£</div>
        <div class="char_block" title="¤">¤</div>
        <div class="char_block" title="¥">¥</div>
        <div class="char_block" title="₹">₹</div>
        <div class="char_block" title="|">|</div>
        <div class="char_block" title="$">$</div>
        <div class="char_block" title="'">'</div>
        <div class="char_block" title="©">©</div>
        <div class="char_block" title="ª">ª</div>
        <div class="char_block" title="«">«</div>
        <div class="char_block" title="¬">¬</div>
        <div class="char_block" title="°">°</div>
        <div class="char_block" title="®">®</div>
        <div class="char_block" title="¯">¯</div>
        <div class="char_block" title="°">°</div>
        <div class="char_block" title="±">±</div>
        <div class="char_block" title="²">²</div>
        <div class="char_block" title="³">³</div>
        <div class="char_block" title="´">´</div>
        <div class="char_block" title="µ">µ</div>
        <div class="char_block" title="¶">¶</div>
        <div class="char_block" title="·">·</div>
        <div class="char_block" title=",">,</div>
        <div class="char_block" title="¹">¹</div>
        <div class="char_block" title="º">º</div>
        <div class="char_block" title="»">»</div>
        <div class="char_block" title="¼">¼</div>

```

```

        <div class="char_block" title="½">½</div>
        <div class="char_block" title="¼">¼</div>
        <div class="char_block" title="¿">¿</div>
        <div class="char_block" title="À">À</div>
        <div class="char_block" title="Á">Á</div>
        <div class="char_block" title="Â">Â</div>
        <div class="char_block" title="Ã">Ã</div>
    </div>
</div>
</div>
<style>
.e-rte-custom-tbar-section #special_char,
.e-rte-custom-tbar-section .char_block {
    display: inline-block;
}
#custom_tbar,
#custom_tbar div{
    cursor: pointer;
}
#rteSpecial_char {
    padding: 15px 0 15px 0;
}
#customTbarDialog {
    max-height: 250px !important;
}
.material .e-rte-custom-tbar-section .char_block.e-active {
    outline: 1px solid #e3165b;
    border-color: #e3165b;
}
.fabric .e-rte-custom-tbar-section .char_block.e-active {
    outline: 1px solid #0078d6;
    border-color: #0078d6;
}
.bootstrap .e-rte-custom-tbar-section .char_block.e-active {
    outline: 1px solid #317ab9;
    border-color: #317ab9;
}
.highcontrast .e-rte-custom-tbar-section .char_block.e-active {
    outline: 1px solid #ffd939;
    border-color: #ffd939;
}
.fabric.e-bigger .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-
btn.e-btn .e-tbar-btn-text,
.highcontrast.e-bigger .e-toolbar .e-toolbar-items .e-toolbar-item .e-
tbar-btn.e-btn .e-tbar-btn-text {
    padding-right: 10px;
}
.bootstrap.e-bigger .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-
btn.e-btn .e-tbar-btn-text,
.bootstrap .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn.e-btn
.e-tbar-btn-text {
    padding-right: 6px;
}
.fabric .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn.e-btn
.e-tbar-btn-text,
.highcontrast .e-toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn.e-
btn .e-tbar-btn-text {

```

```

padding-right: 8px;
}
.e-rte-custom-tbar-section .char_block {
width: 30px;
height: 30px;
line-height: 30px;
margin: 0 5px 5px 0;
text-align: center;
vertical-align: middle;
border: 1px solid #DDDDDD;
font-size: 20px;
cursor: pointer;
user-select: none;
}
#rteSection {
height: 500px;
}
#custom_tbar div{
font-size: 16px;
}
.e-bigger #custom_tbar div{
font-size: 18px;
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The focus will be lost while rendering the required component for the custom toolbar, causing it to render outside the Rich Text Editor and triggering a blur event. During that time, proper functionality will not be achievable. Therefore, it is recommended to set the cssClass property or class as `e-rte-elements` in the dependency component.

Quick inline toolbar

Quick commands are opened as context-menu on clicking the corresponding element. The commands must be passed as string collection to image, text, and link attributes of the [quickToolbarSettings](#) property.

| Target Element | Default Quick Toolbar items |

|-----|-----|

| image | 'Replace', 'Align', 'Caption', 'Remove', 'InsertLink', 'Display', 'AltText', 'Dimension'. |

| link | 'Open', 'Edit', 'UnLink'. |

| text | null
 (Any toolbar [items](#) in the Rich Text Editor can be configured here). |

| table | 'TableHeader', 'TableRows', 'TableColumns', 'BackgroundColor', 'TableRemove', 'Alignments', 'TableCellVerticalAlign', 'Styles'. |

Custom tool can be added to the corresponding quick toolbar, using [quickToolbarSettings](#) property.

The following sample demonstrates the option to insert the image to the Rich Text Editor content as well as option to rotate the image through the quick toolbar. The image rotation functionalities have been achieved through the [toolbarClick](#) event.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
  height: 340,
  quickToolbarSettings: {
    image: [
      'Replace', 'Align', 'Caption', 'Remove', 'InsertLink',
      'OpenImageLink', '-',
      'EditImageLink', 'RemoveImageLink', 'Display', 'AltText',
      'Dimension'
    ]
  },
  toolbarSettings: {
    items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
      'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
      'LowerCase', 'UpperCase', '|',
      'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
      'Outdent', 'Indent', '|',
      'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
      'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
  }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Rich Text Editor features are segregated into individual feature-wise modules. To use quick toolbar, inject the quick toolbar module using the `RichTextEditor.Inject(Image, Link)`.

See Also

- [How to render the toolbar in inline mode](#)
- [How to customize the toolbar items shortcut key](#)

Inline mode in EJ2 JavaScript Rich text editor control

This is the inline example for the Rich Text Editor. For this you must set the [inlineMode](#) property.

Inline edition allows you to select any editable element or click the element on the page and edit it in-place.

Inline editing is a true WYSIWYG formation and on the contrary to Rich Text Editor HTML/MD editing, the styles that are used for edited content comes directly from the document stylesheet. This means that inline editors ignore the default Rich Text Editor content styles.

Show on select/click

Enabling the [onSelection](#) option of [inlineMode](#) makes the inline Rich Text Editor to appear. You can select the text in the editable area otherwise the inline Rich Text Editor will be appear once click into the editable area.

INDEX.TS

```
import { RichTextEditor, Toolbar, HtmlEditor, QuickToolbar } from
 '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
  inlineMode: {
    enable: true,
    onSelection: true
  }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The sample is configured with inline mode of
editor. Initially, the editor is rendered without a <a
href="https://ej2.syncfusion.com/home/" target="_blank">toolbar</a>. The
toolbar becomes visible only when the content is selected/clicked.</p>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to edit the quick toolbar settings](#)
- [How to insert link editing option in the toolbar items](#)
- [How to insert image editing option in the toolbar items](#)

Paste cleanup in EJ2 JavaScript Rich text editor control

The Rich Text Editor allows you to reduce the effort while converting the Microsoft Word content to HTML format with format and styles.

MS Word to HTML

By default, Rich Text Editor consider the following processes on paste content from Microsoft Word.

List conversion: The list elements copied from the Microsoft Word document contains paragraph tags with styles and classes. The list elements are converted to standard HTML list elements by referring the styles and class names in the paragraph tags.

Converting style: The styles of the elements copied from the Microsoft Word document are converted to standard CSS styles and added as inline styles for each respective element.

Tags and comments: The Microsoft Word specific XML tags and comments are removed when cleanup on paste.

Paste cleanup

You can control the formatting and styles on pasting the content to the editor using the `pasteCleanup` settings property. The following settings are available to clean up the content:

API	Description	Default Value	Type
<code>prompt</code>	To invoke prompt dialog with paste options on pasting the content in editor.	false	boolean
<code>plainText</code>	To paste the content as plain text.	false	boolean
<code>keepFormat</code>	To keep the same format with copied content.	true	boolean
<code>deniedTags</code>	To ignore the tags when pasting HTML content.	null	string[]
<code>deniedAttrs</code>	To paste the content by filtering out these attributes from the content.	null	string[]
<code>allowedStyleProps</code>	To paste the content by accepting these style attributes and removing other style attributes.	['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width']	string[]

Rich Text Editor features are segregated into individual feature-wise modules. To use paste cleanup, inject paste cleanup module using the `RichTextEditor.Inject(PasteCleanup)`.

Prompt dialog

When `prompt` is set to true, pasting the content in the editor will open a dialog box that contains three options `Keep`, `Clean`, and `Plain Text` as radio buttons:

1. **Keep**: Radio button to keep the same format with copied content.
2. **Clean**: Radio button to clear all the style formats with copied content.
3. **Plain Text**: Radio button to paste the copied content as plain text without any formatting or style (including the removal of all tags).

When `prompt` value is set true, the API properties `plainText` and `keepFormat` will not be considered for processing when pasting the content.

Paste as plain text

When `plainText` is set to true, the copied content will be converted as plain text by removing all the HTML tags and styles applied to it and only the plain text is pasted in the editor.

When `plainText` value is set true, the API property `prompt` should be set to false, and `keepFormat` will not be considered for processing when pasting the content.

Keep format

When `keepFormat` is set to true, the copied content will maintain all the style formatting allowed in the `allowedStyleProps` on pasting the content in the editor.

When `keepFormat` is set to false, the style in the copied content will be removed without considering the allowed styles in the `allowedStyleProps` when pasting the content in the editor.

When `keepFormat` value is set true, the API property [prompt](#) and [plainText](#) should be set to false.

Denied tags

When `deniedTags` values are set, the tags that matches the 'denied tags' list will be removed on pasting the copied content in the editor. For Example,

1. `'a'`: Paste the content by filtering out anchor tags.
2. `'a[!href]'`: Paste the content by filtering out anchor tags that do not have the 'href' attribute.
3. `'a[href, target]'`: Paste the content by filtering out anchor tags that have the 'href' and 'target' attributes.

Denied attributes

When the `deniedAttrs` values are set, the attributes that matches the 'denied attributes' list will be removed on pasting the copied content in the editor. For Example,

`'id', 'title'`: This will remove the attributes 'id' and 'title' from all tags.

Allowed style properties

By default, the following basic styles are allowed on pasting the content to the editor.

`['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width']`

When you configure `allowedStyleProps`, the styles, which matches the 'allowed style properties' list are allowed, all other style properties will be removed on pasting the content in the editor.

For Example,

`allowedStyleProps: ['color', 'margin']`: This will allow only the style properties 'color' and 'margin' in each pasted element.

In the following example, the paste cleanup related settings are explained with its module configuration

INDEX.TS

```
import { RichTextEditor, Toolbar, HtmlEditor, PasteCleanup } from
 '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, HtmlEditor, PasteCleanup);
let defaultRTE: RichTextEditor = new RichTextEditor({
  value: `<p>Rich Text Editor is a WYSIWYG editing control which will
  reduce the effort for users while trying to express their formatting word
  content as HTML or Markdown format.</p>
    <p><b>Paste Cleanup properties:</b></p>
    <ul>
      <li>
```

```

        <p>prompt - specifies whether to enable the prompt when
pasting in Rich Text Editor.</p>
    </li>
    <li>
        <p>plainText - specifies whether to paste as plain text or
not in Rich Text Editor.</p>
    </li>
    <li>
        <p>keepFormat- specifies whether to keep or remove the
format when pasting in Rich Text Editor.</p>
    </li>
    <li>
        <p>deniedTags - specifies the tags to restrict when pasting
in Rich Text Editor.</p>
    </li>
    <li>
        <p>deniedAttributes - specifies the attributes to restrict
when pasting in Rich Text Editor.</p>
    </li>
    <li>
        <p>allowedStyleProperties - specifies the allowed style
properties when pasting in Rich Text Editor.</p>
    </li>
</ul>`,
    toolbarSettings: {
        type: 'Expand'
    },
    pasteCleanupSettings: {
        prompt: true,
        plainText: false,
        keepFormat: false,
        deniedTags: ['a'],
        deniedAttrs: ['class', 'title', 'id'],
        allowedStyleProps: ['color', 'margin', 'font-size']
    }
});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: The Paste toolbar button action will not work and the clipboard will be restricted due to some limitations in the browser. Pasting copied contents into the Rich Text Editor can be done by using the Ctrl+V keyboard command.

Mention integration in EJ2 JavaScript Rich text editor control

By integrating the [Mention](#) control with a Rich Text Editor, users can easily mention or tag other users or objects from the suggested list without having to manually type out their names or other identifying information.

The [target](#) property of the Mention control allows you to specify the ID of the content editable div element within the Rich Text Editor that you want to bind the Mention control to. This allows you to enable the Mention functionality within the Rich Text Editor, so that users can mention or tag other users or objects from the suggested list while editing the text.

When the user types the @ symbol followed by a character, the Rich Text Editor will display a list of suggestions for items that the user can select from. The user can then select an item from the list by clicking on it, or by typing the name of the item they want to tag.

In the following sample, configured the following properties with popup dimensions.

- [allowSpaces](#) - Allow to continue search action if user enter space after mention character while searching.
- [suggestionCount](#) - The maximum number of items that will be displayed in the suggestion list.
- [itemTemplate](#) - Used to display the customized appearance in suggestion list.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
import { Mention } from '@syncfusion/ej2-dropdowns';
let emailData: { [key: string]: Object }[] = [
    { Name: "Selma Rose", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/2.png",
EmailId: "selma@gmail.com" },
    { Name: "Maria", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/1.png",
EmailId: "maria@gmail.com" },
    { Name: "Russo Kay", Status: "busy", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/8.png",
EmailId: "russo@gmail.com" },
    { Name: "Camden Kate", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/9.png",
EmailId: "camden@gmail.com" },
    { Name: "Robert", Status: "busy", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/dp.png",
EmailId: "robert@gmail.com" },
    { Name: "Garth", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/7.png",
EmailId: "garth@gmail.com" },
    { Name: "Andrew James", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/pic04.png",
EmailId: "andrew@gmail.com" },
    { Name: "Olivia", Status: "busy", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/5.png",
EmailId: "olivia@gmail.com" },
    { Name: "Sophia", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/6.png",
EmailId: "sophia@gmail.com" },
    { Name: "Margaret", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/3.png",
EmailId: "margaret@gmail.com" },
    { Name: "Ursula Ann", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/dp.png",
EmailId: "ursula@gmail.com" },
    { Name: "Laura Grace", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/4.png",
EmailId: "laura@gmail.com" },
    { Name: "Albert", Status: "active", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/pic03.png",
EmailId: "albert@gmail.com" },
    { Name: "William", Status: "away", EmployeeImage:
"https://ej2.syncfusion.com/demos/src/rich-text-editor/images/8.png",
EmailId: "william@gmail.com" }
];
let defaultRTE: RichTextEditor = new RichTextEditor({
    placeholder: 'Type @ and tag the name',
    actionBegin: (args) => {
        if (args.requestType === 'EnterAction') {
            args.cancel = true;
        }
    }
});

```

```

    });
    defaultRTE.appendTo('#mention_integration');
    // Initialize Mention control.
    let emailObj: Mention = new Mention({
        dataSource: emailData,
        fields: { text: 'Name' },
        suggestionCount: 8,
        displayTemplate: '<a href=mailto:${EmailId}
title=${EmailId}>@${Name}</a>',
        itemTemplate: '<table><tr><td><div id="mention-TemplateList"><span
class="e-badge e-badge-success e-badge-overlap e-badge-dot e-badge-bottom
${Status}></span></div></td><td><span class="person">${Name}</span><span
class="email">${EmailId}</span></td></tr></table>',
        popupWidth: '250px',
        popupHeight: '200px',
        target: '#mention_integration_rte-edit-view',
        allowSpaces: true
    });
    emailObj.appendTo('#mentionEditor');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <div id="mentionEditor"></div>
  <div id="mention_integration">
    <p>Hello <span contenteditable="false" class="e-mention-
chip"><a href="mailto:maria@gmail.com"
title="maria@gmail.com">@Maria</a></span>,</p>
    <p>Welcome to the mention integration with rich text editor
demo. Type <code>@</code> character and tag user from the suggestion list.
</p>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[View Sample](#)

See Also

- [Mention](#)

Enter key in EJ2 JavaScript Rich text editor control

Rich Text Editor allows to customize the tag that is inserted when pressing the enter key and shift + enter key in the Rich Text Editor.

Enter key customization

By default, the `<p>` tag will be created while pressing the enter key. The enter key can be customized by using the [enterKey](#) property, where the possible tags that can be used to customize are `<p>`, `<div>`, and `
`.

When the enter key is customized with any of the possible values, pressing the enter key in the editor will create a new tag that is configured. Also, when the enter key is configured the default value of the Rich Text Editor will also change respectively with the configured values.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
  value: `<p>In Rich text Editor, the enter key and shift + enter key
actions can be customized using the enterKey and shiftEnterKey APIs. And the
possible values are as follows:</p><ul><li>P - When 'P' is configured,
pressing enter or shift + enter will create a 'p' tag</li><li>DIV - When
'DIV' is configured, pressing enter or shift + enter will create a 'div'

```

```

tag</li><li>BR - When 'BR' is configured, pressing enter or shift + enter
will create a 'br' tag</li></ul>`,
    height: 250
});
defaultRTE.appendTo('#defaultRTE');
let enterListObj: DropDownList = new DropDownList({
    placeholder: 'When pressing the enter key',
    floatLabelType: 'Always',
    change: (args: any) => {
        if (enterListObj.value === 'DIV') {
            defaultRTE.enterKey = 'DIV';
            defaultRTE.value = `<div>In Rich text Editor, the enter key and
shift + enter key actions can be customized using the enterKey and
shiftEnterKey APIs. And the possible values are as follows:</div><ul><li>P -
When 'P' is configured, pressing enter or shift + enter will create a 'p'
tag</li><li>DIV - When 'DIV' is configured, pressing enter or shift + enter
will create a 'div' tag</li><li>BR - When 'BR' is configured, pressing enter
or shift + enter will create a 'br' tag</li></ul>`;
        } else if (enterListObj.value === 'BR') {
            defaultRTE.enterKey = 'BR';
            defaultRTE.value = `In Rich text Editor, the enter key and shift
+ enter key actions can be customized using the enterKey and shiftEnterKey
APIs. And the possible values are as follows:<ul><li>P - When 'P' is
configured, pressing enter or shift + enter will create a 'p'
tag</li><li>DIV - When 'DIV' is configured, pressing enter or shift + enter
will create a 'div' tag</li><li>BR - When 'BR' is configured, pressing enter
or shift + enter will create a 'br' tag</li></ul>`;
        } else {
            defaultRTE.enterKey = 'P';
            defaultRTE.value = `<p>In Rich text Editor, the enter key and
shift + enter key actions can be customized using the enterKey and
shiftEnterKey APIs. And the possible values are as follows:</p><ul><li>P -
When 'P' is configured, pressing enter or shift + enter will create a 'p'
tag</li><li>DIV - When 'DIV' is configured, pressing enter or shift + enter
will create a 'div' tag</li><li>BR - When 'BR' is configured, pressing enter
or shift + enter will create a 'br' tag</li></ul>`;
        }
    }
});
enterListObj.appendTo('#enterOption');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="default-section">
            <table class="api">
                <tbody>
                    <tr>
                        <td>
                            <div>
                                <select id="enterOption">
                                    <option value="P">Create a new
&#60;p&#62;</option>
                                    <option value="DIV">Create a new
&#60;div&#62;</option>
                                    <option value="BR">Create a new
&#60;br&#62;</option>
                                </select>
                            </div>
                        </td>
                    </tr>
                </tbody>
            </table>
            <br>
            <div id="defaultRTE">
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Shift-Enter key customization

By default, the `
` tag will be created while pressing the shift + enter key. The shift + enter key can be customized by using the [shiftEnterKey](#) property where the possible tags that can be used to customize are `
`, `<p>`, `<div>`.

When the shift + enter key is customized with any of the possible values, pressing the shift + enter key in the editor will create a new tag that is configured. Also, when the shift + enter key is configured the default value of the Rich Text Editor will change respectively with the configured values.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    value: `<p>In Rich text Editor, the enter key and shift + enter key
    actions can be customized using the enterKey and shiftEnterKey APIs. And the
    possible values are as follows:</p><ul><li>P - When 'P' is configured,
    pressing enter or shift + enter will create a 'p' tag</li><li>DIV - When
    'DIV' is configured, pressing enter or shift + enter will create a 'div'
    tag</li><li>BR - When 'BR' is configured, pressing enter or shift + enter
    will create a 'br' tag</li></ul>`,
    height: 250
});
defaultRTE.appendTo('#defaultRTE');
let shiftEnterlistObj: DropDownList = new DropDownList({
    placeholder: 'When pressing the shift + enter key',
    floatLabelType: 'Always',
    change: (args: any) => {
        if (shiftEnterlistObj.value === 'DIV') {
            defaultRTE.shiftEnterKey = 'DIV'
        } else if (shiftEnterlistObj.value === 'P') {
            defaultRTE.shiftEnterKey = 'P'
        } else {
            defaultRTE.shiftEnterKey = 'BR'
        }
    }
});
shiftEnterlistObj.appendTo('#shiftEnterOption');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    richtexteditor/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="api">
            <tbody>
                <tr>
                    <td>
                        <div>
                            <select id="shiftEnterOption">
                                <option value="BR">Create a new
                                &#60;br&#62;</option>
                                <option value="P">Create a new
                                &#60;p&#62;</option>
                                <option value="DIV">Create a new
                                &#60;div&#62;</option>
                            </select>
                        </div>
                    </td>
                </tr>
            </tbody>
        </table>
        <br>
        <div id="defaultRTE">
            <div>
                <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
                </script>
<script src="index.js" type="text/javascript"></script>
            </div>
        </div>
    </body></html>

```

Styling in EJ2 JavaScript Rich text editor control

Font name and size

By default, the editor is initialized with font name and font size as **null**. To change it, select a different font name and font size from the drop-down in the editor's toolbar.

To apply different font style for section of the content, select the text that you would like to change, and select a required font style from the drop-down to apply the changes to the selected text.

FontName DropDowns

The following table list the default font name and width of the [fontname](#) dropdown and available list of font names.

Default Key	Default Value
-----	-----
font name	null
width	65px
items	{ text: 'Segoe UI', value: 'Segoe UI' }, { text: 'Arial', value: 'Arial,Helvetica,sans-serif' }, { text: 'Courier New', value: 'Courier New,Courier,monospace' }, { text: 'Georgia', value: 'Georgia,serif' }, { text: 'Impact', value: 'Impact,Charcoal,sans-serif' }, { text: 'Lucida Console', value: 'Lucida Console,Monaco,monospace' }, { text: 'Tahoma', value: 'Tahoma,Geneva,sans-serif' }, { text: 'Times New Roman', value: 'Times New Roman,Times,serif' }, { text: 'Trebuchet MS', value: 'Trebuchet MS,Helvetica,sans-serif' }, { text: 'Verdana', value: 'Verdana,Geneva,sans-serif' }

FontSize DropDowns

The following table list the default font size and width of the [fontsize](#) dropdown and available list of font size.

Default Key	Default Value
-----	-----
font size	null
width	35px.
items	{ text: '8', value: '8pt' }, { text: '10', value: '10pt' }, { text: '12', value: '12pt' }, { text: '14', value: '14pt' }, { text: '18', value: '18pt' }, { text: '24', value: '24pt' }, { text: '36', value: '36pt' }.

The following sample demonstrates the option to add the font name and font size tools to the toolbar as well as modify the default [width](#) of the tools.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    toolbarSettings: {
```

```

        items: [ 'FontName', 'FontSize' ]
    },
    fontSize: {
        width: '40px'
    },
    fontFamily: {
        width: '60px'
    }
});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>

```

```

        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom font and size

Rich Text Editor support to provide custom fonts and size with existing list. If you want to add additional font names and font sizes to font drop-down, pass the font information as JSON data to the items field of [fontsize](#) and [fontFamily](#) property.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Image, Link, HtmlEditor, QuickToolbar,
NodeSelection } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Image, Link, HtmlEditor, QuickToolbar );
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    toolbarSettings: {
        items: [ 'FontName', 'FontSize' ]
    },
    fontSize: {
        default: '10 pt',
        width: '40px',
        items: [{ text: '8 pt', value: '8pt' },
        { text: '10 pt', value: '10pt' },
        { text: '12 pt', value: '12pt' },
        { text: '14 pt', value: '14pt' },
        { text: '42 pt', value: '42pt' } ]
    },
    fontFamily: {
        default: 'Segoe UI',
        width: '60px',
        items: [
            { text: 'Segoe UI', value: 'Segoe UI' },

```



```

        { text: 'Arial', value: 'Arial,Helvetica,sans-serif' },
        { text: 'Courier New', value: 'Courier New,Courier,monospace' },
        { text: 'Georgia', value: 'Georgia,serif' },
        { text: 'Impact', value: 'Impact,Charcoal,sans-serif' },
        { text: 'Calibri Light', value: 'CalibriLight' }]
    }
});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
      <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
      <li><p>Provides a fully customizable toolbar.</p></li>

```

```

        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Font and Background color

To apply font color or background color for a selected content of Rich Text Editor, use font color and background color tools.

Rich Text Editor support to provide customs font color and background color with existing list through the [colorCode](#) field of [fontColor](#) and [backgroundColor](#).

The FontColor and BackgroundColor property has two [mode](#) Picker and Palette. Palette mode has predefined set of [colorCode](#) and in the picker mode, more colors has been provided. Through [modeSwitcher](#), you can able to switch between these two options.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    toolbarSettings: {
        items: [ 'FontColor', 'BackgroundColor' ]
    },
    backgroundColor: {
        modeSwitcher : true
    },
    fontColor: {
        modeSwitcher : true
    }
});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```
}  
    </script>  
<script src="index.js" type="text/javascript"></script>  
</body></html>
```

Editor content styles

By default, The content styles of Rich Text Editor are not returned while retrieving HTML value from the editor. So, the styles are not applied when using the HTML value outside of the editor. To get the styles to Rich Text Editor's content for your application, You can copy and use the below styles directly in your application. The styles listed below which used in the UI elements of the Rich Text Editor.

Make sure to add a CSS class 'e-rte-content' to the content container.

```
,  
.e-rte-content p {  
margin: 0 0 10px;  
margin-bottom: 10px;  
}  
.e-rte-content li {  
margin-bottom: 10px;  
}  
.e-rte-content h1 {  
font-size: 2.17em;  
font-weight: 400;  
line-height: 1;  
margin: 10px 0;  
}  
.e-rte-content h2 {  
font-size: 1.74em;  
font-weight: 400;  
margin: 10px 0;  
}  
.e-rte-content h3 {  
font-size: 1.31em;  
font-weight: 400;  
margin: 10px 0;  
}  
.e-rte-content h4 {
```

```
font-size: 1em;
font-weight: 400;
margin: 0;
}
.e-rte-content h5 {
font-size: 00.8em;
font-weight: 400;
margin: 0;
}
.e-rte-content h6 {
font-size: 00.65em;
font-weight: 400;
margin: 0;
}
.e-rte-content blockquote {
margin: 10px 0;
margin-left: 0;
padding-left: 5px;
}
.e-rte-content pre {
background-color: inherit;
border: 0;
border-radius: 0;
color: #333;
font-size: inherit;
line-height: inherit;
margin: 0 0 10px;
overflow: visible;
padding: 0;
white-space: pre-wrap;
word-break: inherit;
word-wrap: break-word;
}
```

```
.e-rte-content strong, .e-rte-content b {  
font-weight: 700;  
}  
.e-rte-content a {  
text-decoration: none;  
-webkit-user-select: auto;  
-ms-user-select: auto;  
user-select: auto;  
}  
.e-rte-content a:hover {  
text-decoration: underline;  
}  
.e-rte-content h3 + h4,  
.e-rte-content h4 + h5,  
.e-rte-content h5 + h6 {  
margin-top: 0.6em;  
}  
.e-rte-content .e-rte-image.e-imgbreak {  
border: 0;  
cursor: pointer;  
display: block;  
float: none;  
margin: 5px auto;  
max-width: 100%;  
position: relative;  
}  
.e-rte-content .e-rte-image {  
border: 0;  
cursor: pointer;  
display: block;  
float: none;  
margin: auto;  
max-width: 100%;
```

```
position: relative;
}

.e-rte-content .e-rte-image.e-imginline {
display: inline-block;
float: none;
margin-left: 5px;
margin-right: 5px;
max-width: calc(100% - (2 * 5px));
vertical-align: bottom;
}

.e-rte-content .e-rte-image.e-imgcenter {
cursor: pointer;
display: block;
float: none;
margin: 5px auto;
max-width: 100%;
position: relative;
}

.e-rte-content .e-rte-image.e-imgleft {
float: left;
margin: 0 5px 0 0;
text-align: left;
}

.e-rte-content .e-rte-image.e-imgright {
float: right;
margin: 0 0 0 5px;
text-align: right;
}

.e-rte-content .e-rte-img-caption {
display: inline-block;
margin: 5px auto;
max-width: 100%;
position: relative;
```

```
}  
.e-rte-content .e-rte-img-caption.e-caption-inline {  
display: inline-block;  
margin: 5px auto;  
margin-left: 5px;  
margin-right: 5px;  
max-width: calc(100% - (2 * 5px));  
position: relative;  
text-align: center;  
vertical-align: bottom;  
}  
.e-rte-content .e-rte-img-caption.e-imgcenter {  
display: block;  
}  
.e-rte-content .e-rte-img-caption .e-rte-image.e-imgright,  
.e-rte-content .e-rte-img-caption .e-rte-image.e-imleft {  
float: none;  
margin: 0;  
}  
.e-rte-content .e-rte-table {  
border-collapse: collapse;  
empty-cells: show;  
}  
.e-rte-content .e-rte-table td,  
.e-rte-content .e-rte-table th {  
border: 1px solid #bdbdbd;  
height: 20px;  
min-width: 20px;  
padding: 2px 5px;  
vertical-align: middle;  
}  
.e-rte-content .e-rte-table.e-dashed-border td,  
.e-rte-content .e-rte-table.e-dashed-border th {
```



```
border-style: dashed;
}
.e-rte-content .e-rte-img-caption .e-img-inner {
box-sizing: border-box;
display: block;
font-size: 16px;
font-weight: initial;
margin: auto;
opacity: .9;
position: relative;
text-align: center;
width: 100%;
}
.e-rte-content .e-rte-img-caption .e-img-wrap {
display: inline-block;
margin: auto;
padding: 0;
width: 100%;
}
.e-rte-content blockquote {
border-left: solid 2px #333;
}
.e-rte-content a {
color: #2e2ef1;
}
.e-rte-content .e-rte-table th {
background-color: #e0e0e0;
}
,
```

See Also

- [How to add google fonts to the font family](#)
- [How to customize the toolbar items shortcut key](#)
- [How to customize the placeholder](#)
- [How to change the default font family](#)

Image in EJ2 JavaScript Rich text editor control

Rich Text Editor allows to insert images from online source as well as local computer where you want to insert the image in your content. For inserting the image to the Rich Text Editor, the following list of options have been provided in the [insertImageSettings](#)

Options	Description
----- -----	
allowedTypes	Specifies the extensions of the image types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as .jpg and .png.
display	Sets the default display for an image when it is inserted in to the Rich Text Editor. Possible options are: 'inline' and 'break'.
width	Sets the default width of the image when it is inserted in the Rich Text Editor.
height	Sets the default height of the image when it is inserted in the Rich Text Editor.
saveUrl	Provides URL to map the action result method to save the image.
path	Specifies the location to store the image.

Rich Text Editor features are segregated into individual feature-wise modules. To use image and link tool, inject image module using the `RichTextEditor.Inject(Image)`.

Upload options

Through the `browse` option, select the image from the local machine and insert into the Rich Text Editor content.

If the path field is not specified in the [insertImageSettings](#), the image will be transferred into base 64 and blob url for the image will be created and the generated url will be set to the src property of `` tag.

`ts

```

```

,

If you want to insert a lot of tiny images in the editor and don't want a specific physical location for saving images, you can opt to save format as Base64.

In the following sample, the image has been loaded from the local machine and it will be saved in the given location.

INDEX.TS

```
import { RichTextEditor, Toolbar, Image, Link, HtmlEditor, QuickToolbar,
NodeSelection } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Image, Link, HtmlEditor, QuickToolbar );
let defaultRTE: RichTextEditor = new RichTextEditor({
  toolbarSettings: {
    items: ['Image']}
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
      <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
      <li><p>Provides a fully customizable toolbar.</p></li>
      <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
      <li><p>Supports third-party library integration.</p></li>
      <li><p>Allows preview of modified content before saving
it.</p></li>
      <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
      </ul>
    </div>
  </div>
</body>
</html>

```

```

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

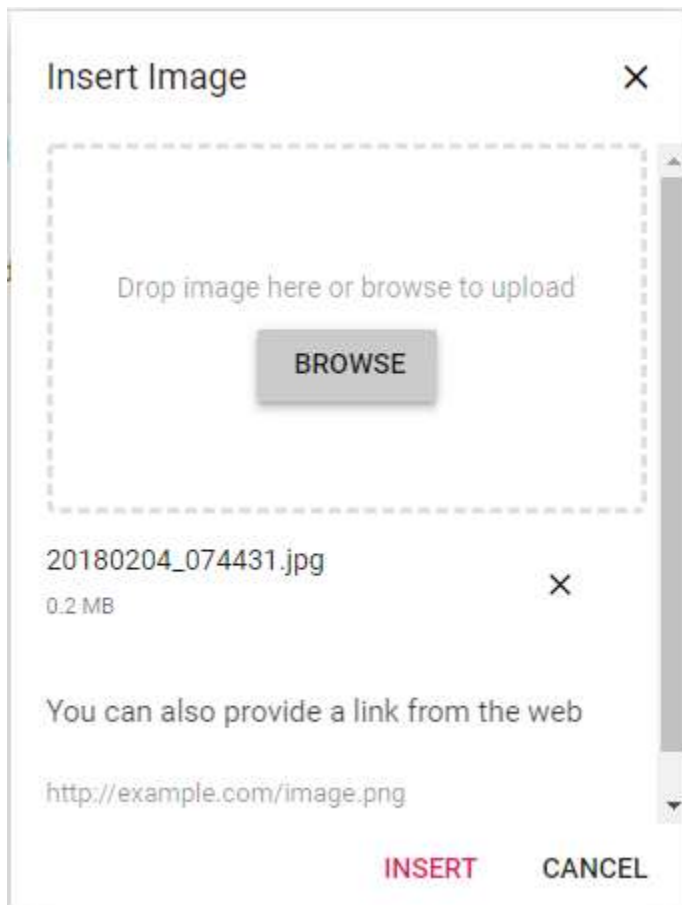
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Image delete

To remove an image from the Rich Text Editor content, select the image and click **Remove** tool from the quick toolbar. It will delete the image from the Rich Text Editor content as well as from the service location if the `removeUrl` is given.

Once you select the image from the local machine, the URL for the image will be generate. From there, you can remove the image from the service location by clicking the cross icon.



The following sample explains, how to configure `removeUrl` to remove a saved image from the remote service location, when the following image remove actions are performed:

- `delete` key action.
- `backspace` key action.
- Removing uploaded image file from the insert image dialog.
- Deleting image using the quick toolbar `remove` option.

INDEX.TS

```
import { RichTextEditor, HtmlEditor, Toolbar, QuickToolbar, Image,
FileManager } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(HtmlEditor, Toolbar, QuickToolbar, Image,
FileManager);
let defaultRTE: RichTextEditor = new RichTextEditor({
  insertImageSettings: {
    saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
    removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
  },
  toolbarSettings: {
    items: ['Image']
  }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
```

```

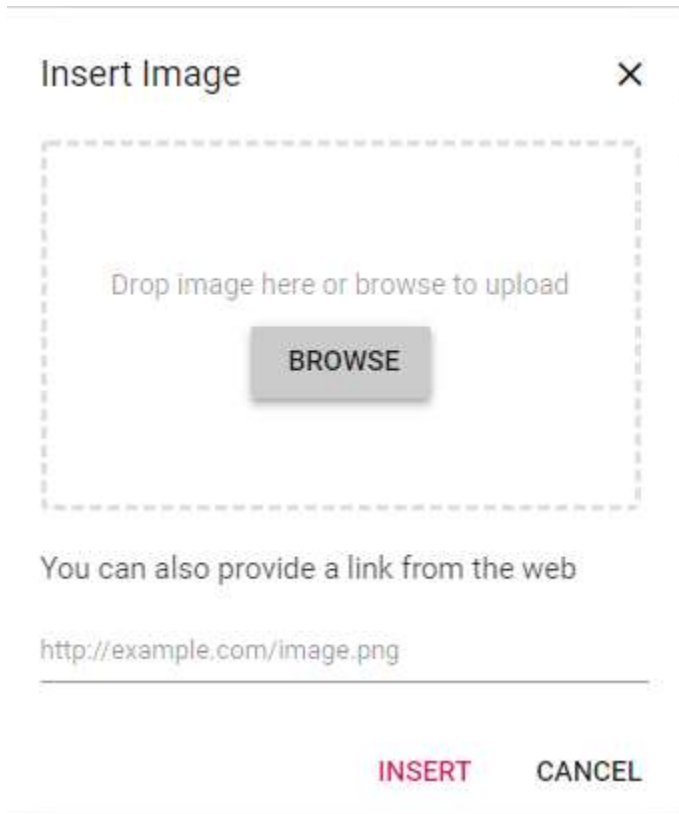
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul>
                <li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
                <li><p>Capable of handling markdown editing.</p></li>
                <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
                <li><p>Provides a fully customizable toolbar.</p></li>
                <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
                <li><p>Supports third-party library integration.</p></li>
                <li><p>Allows preview of modified content before saving
it.</p></li>
                <li><p>Handles images, hyperlinks, video, hyperlinks,
uploads, etc.</p></li>
            </ul>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Insert from web

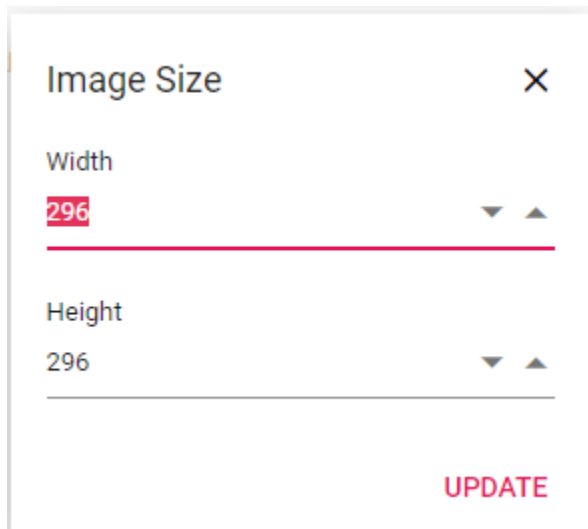
To insert an image from the online source like Google, Ping, etc., you should enable the image tool on the editor's toolbar. By default, the image tool opens a simple dialog which allows you to insert an image from online source.



Dimension

Sets the default width and height of the image when it is inserted in the Rich Text Editor using [width](#) and [height](#) of [insertImageSettings](#).

Through the [quickToolbar](#), also you can change the width and height using [Change Size](#) option. Once click into the option. The Image Size dialog will open as below. In that specify the width and height of the image in pixel.



Caption and Alt Text

Image caption and alternative text can be specified for the Inserted image in the Rich Text Editor through the [quickToolbar](#) options such as Image Caption and Alternative Text.

Through the Alternative Text option, set the alternative text for the image, when the image is not upload successfully into the Rich Text Editor.

By clicking the Image Caption, the image will get wrapped in an image element with a caption. Then, you can type caption content inside the Rich Text Editor.

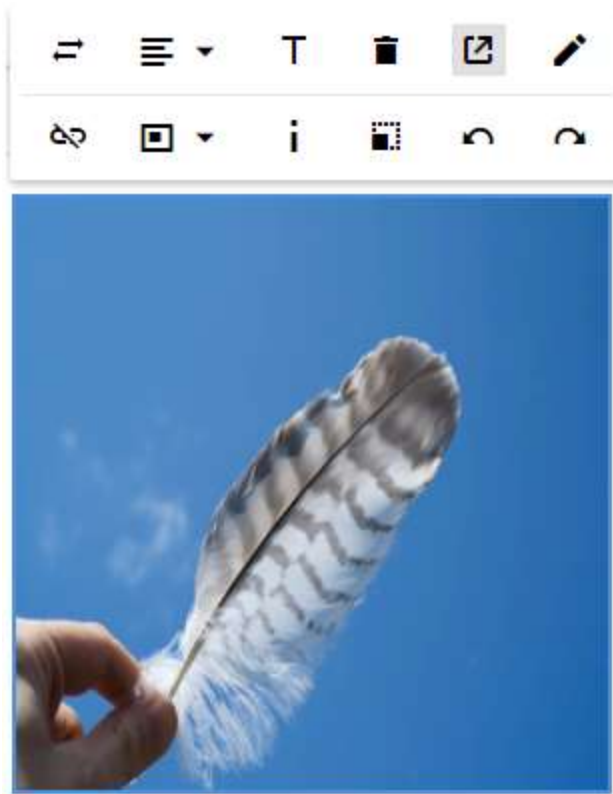
Display position

Sets the default display for an image when it is inserted in the Rich Text Editor using [display](#) field in [insertImageSettings](#). It has two possible options: `inline` and `break`.

```
`ts
let defaultRTE: RichTextEditor = new RichTextEditor({
insertImageSettings: {
display: 'inline'
}
});
defaultRTE.appendTo('#defaultRTE');
```

Image with link

The hyperlink itself can be an image in Rich Text Editor. If the image given as hyperlink, the remove, edit, and open link will be added to the quick toolbar of image. For further details about link, see the [link documentation](#).



Resize

Rich text editor has a built-in image inserting support. The resize points will be appearing on each corner of image when focus. So, users can resize the image using mouse points or thumb through the resize points easily. Also, the resize calculation will be done based on aspect ratio.



Drag and Drop

By default, the Rich Text Editor allows you to insert images by drag-and-drop from the local file system such as Windows Explorer into the content editor area. And, you can upload the images to the server before inserting into the editor by configuring the `saveUrl` property. The images can be repositioned anywhere within the editor area by dragging and dropping the image.

In the following sample, you can see feature demo.

INDEX.TS

```
import { RichTextEditor, Toolbar, Image, Link, HtmlEditor, QuickToolbar,
NodeSelection } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Image, Link, HtmlEditor, QuickToolbar );
let defaultRTE: RichTextEditor = new RichTextEditor({
    insertImageSettings : {
        saveUrl : 'https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save'
    }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
```

```

        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop with specific extension images

You can allow the specific images alone to be uploaded using the the allowedTypes property. By default, the Rich Text Editor allows the JPG, JPEG, and PNG formats. You can configure this formats as follows.

```
`ts
```

```
insertImageSettings: {
```

```
  allowedTypes: ['.jpg']
```

```
}
```

```
,
```

Prevent drag and drop action

You can prevent drag-and-drop action by setting the actionBegin argument cancel value to true. The following code shows how to prevent the drag-and-drop.

```
`ts
```

```
actionBegin: function (args: any): void {
```

```
  if(args.type === 'drop' || args.type === 'dragstart') {
```

```
    args.cancel =true;
```

```
}
```

```
}
,
```

See Also

- [How to edit the quick toolbar settings](#)
- [How to use link editing option in the toolbar items](#)

Audio in EJ2 JavaScript Rich text editor control

The Rich Text Editor allows you to insert audio from online sources and local computers and then insert them into your content. You can insert the audio with the following list of options in the [insertAudioSettings](#) property.

Options	Description
-----	-----
allowedTypes	Specifies the extensions of the audio types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as .mp3, .wav, .m4a and .wma.
layoutOption	Sets the default display for audio when it is inserted into the Rich Text Editor. Possible options are: Inline and Break.
saveFormat	Sets the default save format of the audio element when inserted. Possible options are: Blob and Base64.
saveUrl	Provides URL to map the action result method to save the audio.
removeUrl	Provides URL to map the action result method to remove the audio.
path	Specifies the location to store the audio.

Configure audio tool in the toolbar

You can add an audio tool in the Rich Text Editor toolbar using the [toolbarSettings](#) [items](#) property.

To configure the Audio toolbar item, refer to the below code.

INDEX.JS

```
/**
 * Rich Text Editor - RemoveUrl sample
 */
var defaultRTE = new ej.richtexteditor.RichTextEditor({
  insertAudioSettings: {
    saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
    removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
  },
  toolbarSettings: {
    items: ['Audio']
  }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul>
        <li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
        <li><p>Capable of handling markdown editing.</p></li>
        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks,
uploads, etc.</p></li>
      </ul>
    </div>
  </div>

```

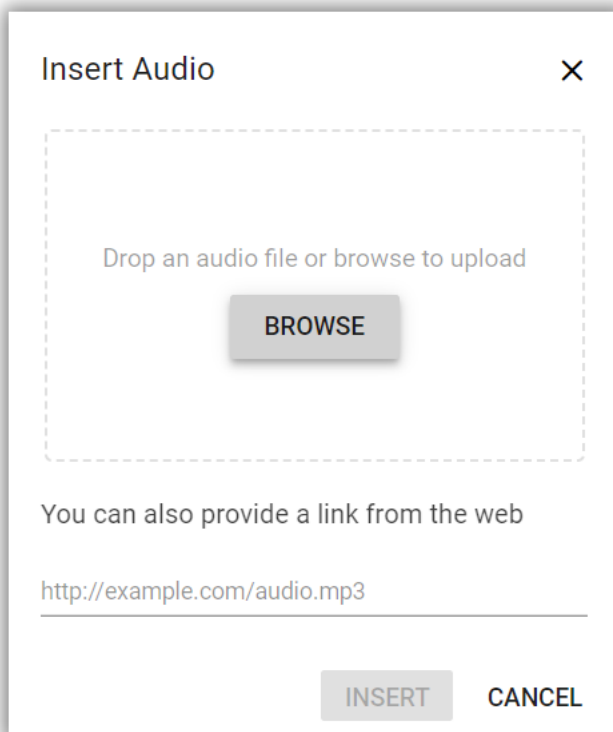
```
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Insert audio from the web

You can insert audio from either the hosted link or the local machine, by clicking the audio button in the editor's toolbar. On clicking the audio button, a dialog opens, which allows you to insert audio from the web URL.

Insert from web URL

By default, the audio tool opens the audio dialog, allowing you to insert audio from an online source. Inserting the URL will be added to the `src` attribute of the `<source>` tag.



Insert audio from local machine

You can use the `browse` option on the audio dialog, to select the audio from the local machine and insert it into the Rich Text Editor content.

If the path field is not specified in the [insertAudioSettings](#), the audio will be converted into the `Blob` URL or `Base64` and inserted inside the Rich Text Editor.

Restrict audio upload based on size

You can restrict the audio uploaded from the local machine when the uploaded audio file size is greater than the allowed size by using the [fileUploading](#) event.

The file size in the argument will be returned in **bytes**.

In the following illustration, the audio size has been validated before uploading, and it is determined whether the audio has been uploaded or not.

```
`ts
<script>
var defaultRTE = new new ej.richtexteditor.RichTextEditor({
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: "https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save",
path: "../Files/"
},
fileUploading: onFileUpload
});
defaultRTE.appendTo('#defaultRTE');
function onFileUpload (args) {
var sizeInBytes = args.fileData.size;
var fileSize = 500000;
if (fileSize < sizeInBytes) {
args.Cancel = true;
}
}
</script>
`
```

Server-side action

The selected audio can be uploaded to the required destination using the controller action below. Map this method name in [insertAudioSettings.saveUrl](#) and provide the required destination path through [insertAudioSettings.path](#) properties.

If you want to insert lower-sized audio files in the editor and don't want a specific physical location for saving the audio, you can opt to save the format as **Base64**.

In the following code blocks, the audio module has been injected and can insert the audio files saved in the specified path.

```
`ts
<script>
var defaultRTE = new ej.richtexteditor.RichTextEditor({
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/SaveFiles",
path: "[SERVICEHOSTEDPATH]/Files/"
}
});
defaultRTE.appendTo('#defaultRTE');
</script>
`

`c#
using System;
using System.IO;
using FileUpload.Models;
using System.Diagnostics;
using System.Net.Http.Headers;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System.Collections.Generic;
using Microsoft.AspNetCore.Hosting;
namespace FileUpload.Controllers
{
public class HomeController : Controller
{
private IHostingEnvironment hostingEnv;
public HomeController(IHostingEnvironment env)
{
```



```
hostingEnv = env;
}
public IActionResult Index()
{
    return View();
}
[AcceptVerbs("Post")]
public void SaveFiles(IList<IFormFile> UploadFiles)
{
    try
    {
        foreach (IFormFile file in UploadFiles)
        {
            if (UploadFiles != null)
            {
                string filename = ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim("");
                filename = hostingEnv.WebRootPath + "\\Files" + $"@\"{filename}\"";
                // Create a new directory, if it does not exists
                if (!Directory.Exists(hostingEnv.WebRootPath + "\\Files"))
                {
                    Directory.CreateDirectory(hostingEnv.WebRootPath + "\\Files");
                }
                if (!System.IO.File.Exists(filename))
                {
                    using (FileStream fs = System.IO.File.Create(filename))
                    {
                        file.CopyTo(fs);
                        fs.Flush();
                    }
                    Response.StatusCode = 200;
                }
            }
        }
    }
}
```

```

}
catch (Exception)
{
    Response.StatusCode = 204;
}
}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}
}
}
,

```

Audio save format

The audio files can be saved as **Blob** or **Base64** URL by using the [insertAudioSettings.saveFormat](#) property, which is of enum type, and the generated URL will be set to the **src** attribute of the **<source>** tag.

The default **saveFormat** property is set to **Blob** format.

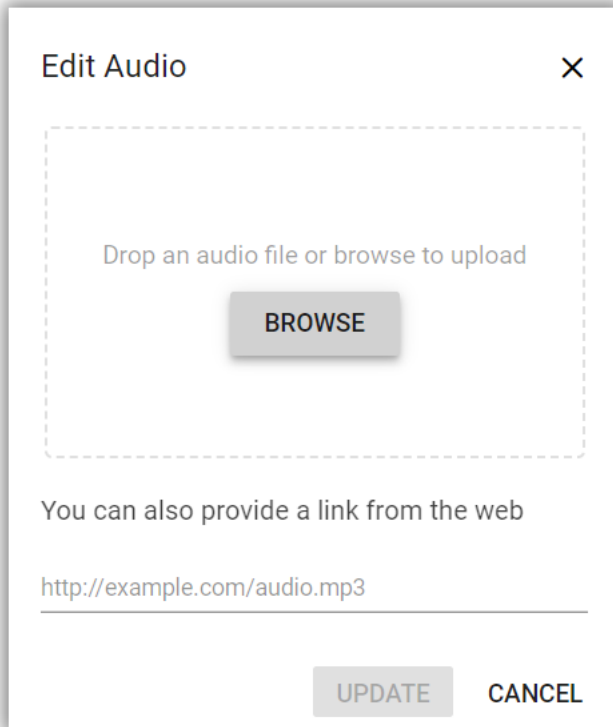
```

`html
<audio>
<source src="blob:http://ej2.syncfusion.com/3ab56a6e-ec0d-490f-85a5-f0aeb0ad8879"
type="audio/mp3" >
</audio>
<audio>
<source src="data:audio/mp3;base64,iVBORw0KGgoAAAANSUHEUgAAADAAAAAwCAYAAABXAvmHA"
type="audio/mp3" >
</audio>
,

```

Replacing audio

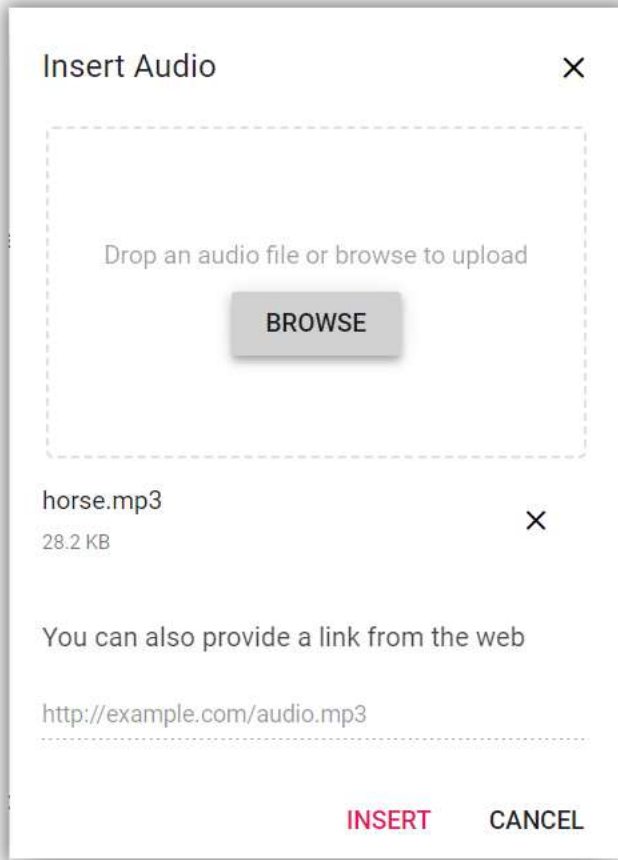
Once an audio file has been inserted, you can change it using the Rich Text Editor [quickToolbarSettings](#) **audioReplace** option. You can replace the audio file using the web URL or the browse option in the audio dialog.



Delete audio

To remove audio from the Rich Text Editor content, select the audio and click the `audioRemove` button from the quick toolbar. It will delete the audio from the Rich Text Editor content as well as from the service location if the [insertAudioSettings.removeUrl](#) is given.

Once you select the audio from the local machine, the URL for the audio will be generated. You can remove the audio from the service location by clicking the cross icon.



Display position

Sets the default display property for audio when it is inserted in the Rich Text Editor using the [insertAudioSettings.layoutOption](#) property. It has two possible options: `Inline` and `Break`. When updating the display positions, it updates the audio elements' layout position.

The default `layoutOption` property is set to `Inline`.

```
`ts
var defaultRTE = new ej.richtexteditor.RichTextEditor({
insertAudioSettings: {
layoutOption: 'Inline'
}
});
defaultRTE.appendTo('#defaultRTE');
```

Rename audio before inserting

You can use the [insertAudioSettings](#) property, to specify the server handler to upload the selected audio. Then by binding the [fileUploadSuccess](#) event, you can receive the modified file name from the server and update it in the Rich Text Editor's insert audio dialog.

```

`html
<div id='defaultRTE'>

<p>The Rich Text Editor is WYSIWYG ("what you see is what you get") editor useful to create and edit
content, and return the valid <a href="https://ej2.syncfusion.com/home/" target="blank">HTML
markup</a> or <a href="https://ej2.syncfusion.com/home/" target="blank">markdown</a> of the
content</p>

</div>
`
`ts
<script>
var defaultRTE = new ej.richtexteditor.RichTextEditor({
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Rename",
path: "[SERVICEHOSTEDPATH]/Files/"
},
fileUploadSuccess: onFileUploadSuccess
});
defaultRTE.appendTo('#defaultRTE');
function onFileUploadSuccess (args) {
alert("Get the new file name here");
if (args.e.currentTarget.getResponseHeader('name') != null) {
args.file.name = args.e.currentTarget.getResponseHeader('name');
var filename = document.querySelectorAll(".e-file-name")[0];
filename.innerHTML = args.file.name.replace(document.querySelectorAll(".e-file-type")[0].innerHTML,
"");
filename.title = args.file.name;
}
}
</script>
`

```

To configure server-side handler, refer to the below code.

```
`c#
int x = 0;
string file;
[AcceptVerbs("Post")]
public void Rename()
{
    try
    {
        var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
        fileName = httpPostedFile.FileName;
        if (httpPostedFile != null)
        {
            var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Files");
            if (!Directory.Exists(fileSave))
            {
                Directory.CreateDirectory(fileSave);
            }
            var fileName = Path.GetFileName(httpPostedFile.FileName);
            var fileSavePath = Path.Combine(fileSave, fileName);
            while (System.IO.File.Exists(fileSavePath))
            {
                fileName = "rteFiles" + x + "-" + fileName;
                fileSavePath = Path.Combine(fileSave, fileName);
                x++;
            }
            if (!System.IO.File.Exists(fileSavePath))
            {
                httpPostedFile.SaveAs(fileSavePath);
                HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
                Response.Clear();
                Response.Headers.Add("name", fileName);
                Response.ContentType = "application/json; charset=utf-8";
                Response.StatusDescription = "File uploaded succesfully";
            }
        }
    }
}
```

```

Response.End();
}
}
}
catch (Exception e)
{
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
    Response.Clear();
    Response.ContentType = "application/json; charset=utf-8";
    Response.StatusCode = 204;
    Response.Status = "204 No Content";
    Response.StatusDescription = e.Message;
    Response.End();
}
}
,

```

Upload audio with authentication

You can add additional data with the audio uploaded from the Rich Text Editor on the client side, which can even be received on the server side by using the [fileUploading](#) event and its `customFormData` argument, you can pass parameters to the controller action. On the server side, you can fetch the custom headers by accessing the form collection from the current request, which retrieves the values sent using the POST method.

By default, it doesn't support the `UseDefaultCredentials` property; we need to manually append the default credentials with the upload request.

```

`ts
<script>
var defaultRTE = new ej.richtexteditor.RichTextEditor({
toolbarSettings: {
items: ['Audio']
},
insertAudioSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/SaveFiles",
path: "[SERVICEHOSTEDPATH]/Files/"
},
fileUploading: onFileUpload

```

```

});
defaultRTE.appendTo('#defaultRTE');
function onFileUpload (args) {
var accessToken = "Authorization_token";
// adding custom Form Data
args.customFormData = [{ 'Authorization': accessToken }];
}
</script>
`c#
public void SaveFiles(IList<IFormFile> UploadFiles)
{
string currentPath = Request.Form["Authorization"].ToString();
}

```

See Also

- [How to edit the quick toolbar settings](#)
- [How to use the link editing option in the toolbar items](#)

Video in EJ2 JavaScript Rich text editor control

The Rich Text Editor allows you to insert videos from online sources and local computers and then insert them into your content. You can insert the video with the following list of options in the [insertVideoSettings](#) property.

Options	Description
-----	-----
allowedTypes	Specifies the extensions of the video types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as <code>.mp4</code> , <code>.mov</code> , <code>.wmv</code> and <code>.avi</code> .
layoutOption	Sets the default display for a video when it is inserted into the Rich Text Editor. Possible options are: <code>Inline</code> and <code>Break</code> .
saveFormat	Sets the default save format of the video element when inserted. Possible options are: <code>Blob</code> and <code>Base64</code> .
width	Sets the default width of the video when it is inserted in the Rich Text Editor.
minWidth	Sets the minWidth of the video element when it is inserted in the Rich Text Editor.
maxWidth	Sets the maxWidth of the video element when it is inserted in the Rich Text Editor.

- | height | Sets the default height of the video when it is inserted in the Rich Text Editor.
- | minHeight | Sets the minHeight of the video element when it is inserted in the Rich Text Editor.
- | maxHeight | Sets the maxHeight of the video element when it is inserted in the Rich Text Editor.
- | saveUrl | Provides URL to map the action result method to save the video.
- | removeUrl | Provides URL to map the action result method to remove the video.
- | path | Specifies the location to store the video.
- | resize | Sets the resizing action for the video element.
- | resizeByPercent | Sets the percentage values for the video element with the resizing action.

Configure the video tool in the toolbar

You can add the **video** tool in the Rich Text Editor toolbar using the **toolbarSettings** [items](#) property.

To configure the **Video** toolbar item, refer to the below code.

INDEX.JS

```
/**
 * Rich Text Editor - RemoveUrl sample
 */
var defaultRTE = new ej.richtexteditor.RichTextEditor({
  insertVideoSettings: {
    saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
    removeUrl:
    'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
  },
  toolbarSettings: {
    items: ['Video']
  }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul>
                <li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
                <li><p>Capable of handling markdown editing.</p></li>
                <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
                <li><p>Provides a fully customizable toolbar.</p></li>
                <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
                <li><p>Supports third-party library integration.</p></li>
                <li><p>Allows preview of modified content before saving
it.</p></li>
                <li><p>Handles images, hyperlinks, video, hyperlinks,
uploads, etc.</p></li>
            </ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

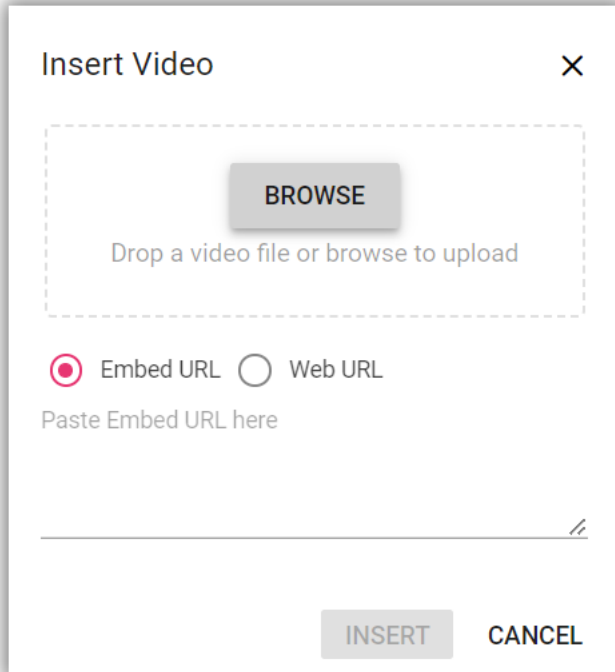
```

Insert a video from the web

You can insert a video from either the hosted link or the local machine by clicking the video button in the editor's toolbar. On Clicking the Video button, a dialog opens which allows you to insert video from the Embedded URL or web URL.

Insert from embed URL

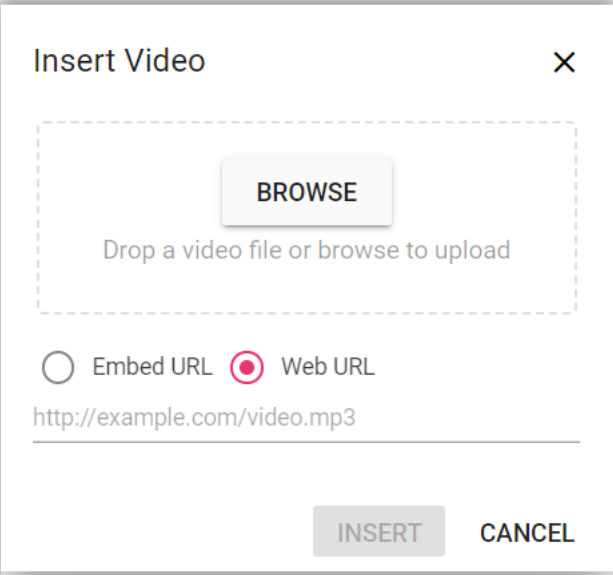
The insert video dialog opens with the **Embed URL** option as default which allows you to insert an embedded URL.



The dialog box is titled "Insert Video" with a close button (X) in the top right corner. It features a dashed rectangular area containing a "BROWSE" button and the text "Drop a video file or browse to upload". Below this, there are two radio buttons: "Embed URL" (which is selected, indicated by a red dot) and "Web URL". Under the "Embed URL" option, there is a text input field with the placeholder text "Paste Embed URL here". At the bottom of the dialog, there are two buttons: "INSERT" and "CANCEL".

Insert from web URL

You can switch to **Web URL** by selecting the web URL check box. Inserting with the web URL option will add the video URL as the `src` attribute of the `<source>` tag.



The dialog box is titled "Insert Video" with a close button (X) in the top right corner. It features a dashed rectangular area containing a "BROWSE" button and the text "Drop a video file or browse to upload". Below this, there are two radio buttons: "Embed URL" and "Web URL" (which is selected, indicated by a red dot). Under the "Web URL" option, there is a text input field containing the example URL "http://example.com/video.mp3". At the bottom of the dialog, there are two buttons: "INSERT" and "CANCEL".

Insert video from local machine

You can use the **browse** option on the video dialog, to select the video from the local machine and insert it into the Rich Text Editor content.

If the path field is not specified in the [insertVideoSettings](#), the video will be converted into the **Blob** URL or **Base64** and inserted inside the Rich Text Editor.

Restrict video upload based on size

You can restrict the video uploaded from the local machine when the uploaded video file size is greater than the allowed size by using the [fileUploading](#) event.

The file size in the argument will be returned in **bytes**.

In the following example, the video size has been validated before uploading and determined whether the video has been uploaded or not.

```
`ts
<script>
var defaultRTE = new ej.richtexteditor.RichTextEditor({
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
saveUrl: "https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save",
path: "../Files/"
},
fileUploading: onFileUpload
});
defaultRTE.appendTo('#defaultRTE');
function onFileUpload (args) {
var sizeInBytes = args.fileData.size;
var fileSize = 500000;
if (fileSize < sizeInBytes) {
args.Cancel = true;
}
}
</script>
`
```

Server-side action

The selected video can be uploaded to the required destination using the controller action below. Map this method name in [insertVideoSettings.saveUrl](#) and provide the required destination path through [insertVideoSettings.path](#) properties.

If you want to insert lower-sized video files in the editor and don't want a specific physical location for saving the video, you can save the format as **Base64**.

In the following code blocks, the video module has been injected and can insert the video files saved in the specified path.

```
`ts
<script>
var defaultRTE = new ej.richtexteditor.RichTextEditor({
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/SaveFiles",
path: "[SERVICEHOSTEDPATH]/Files/"
}
});
defaultRTE.appendTo('#defaultRTE');
</script>
`
```

```
`c#
using System;
using System.IO;
using FileUpload.Models;
using System.Diagnostics;
using System.Net.Http.Headers;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System.Collections.Generic;
using Microsoft.AspNetCore.Hosting;
namespace FileUpload.Controllers
{
public class HomeController : Controller
{
private IHostingEnvironment hostingEnv;
public HomeController(IHostingEnvironment env)
{
```

```
hostingEnv = env;
}
public IActionResult Index()
{
    return View();
}
[AcceptVerbs("Post")]
public void SaveFiles(IList<IFormFile> UploadFiles)
{
    try
    {
        foreach (IFormFile file in UploadFiles)
        {
            if (UploadFiles != null)
            {
                string filename = ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim("");
                filename = hostingEnv.WebRootPath + "\\Files" + $"@\"{filename}\"";
                // Create a new directory, if it does not exists
                if (!Directory.Exists(hostingEnv.WebRootPath + "\\Files"))
                {
                    Directory.CreateDirectory(hostingEnv.WebRootPath + "\\Files");
                }
                if (!System.IO.File.Exists(filename))
                {
                    using (FileStream fs = System.IO.File.Create(filename))
                    {
                        file.CopyTo(fs);
                        fs.Flush();
                    }
                    Response.StatusCode = 200;
                }
            }
        }
    }
}
```

```

}
catch (Exception)
{
    Response.StatusCode = 204;
}
}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}
}
}
,

```

Video save format

The video files can be saved as **Blob** or **Base64** URL by using the [insertVideoSettings.saveFormat](#) property, which is of enum type, and the generated URL will be set to the **src** attribute of the **<source>** tag.

The default **saveFormat** property is set to **Blob** format.

```

`ts
<video>
<source src="blob:http://ej2.syncfusion.com/3ab56a6e-ec0d-490f-85a5-f0aeb0ad8879"
type="video/mp4" >
</video>
<video>
<source src="data:video/mp4;base64,iVBORw0KGgoAAAANSUhEUgAAADAAAAAwCAYAAABXAvmHA"
type="video/mp4" >
</video>
,

```

Replacing video

Once a video file has been inserted, you can replace it using the Rich Text Editor [quickToolbarSettings](#) **videoReplace** option. You can replace the video file either by using the embedded URL or the web URL and the browse option in the video dialog.

Edit Video

BROWSE

Drop a video file or browse to upload

☒ Embed URL ☐ Web URL

Paste Embed URL here

UPDATE

CANCEL

Edit Video

BROWSE

Drop a video file or browse to upload

☐ Embed URL ☒ Web URL

http://example.com/video.mp3

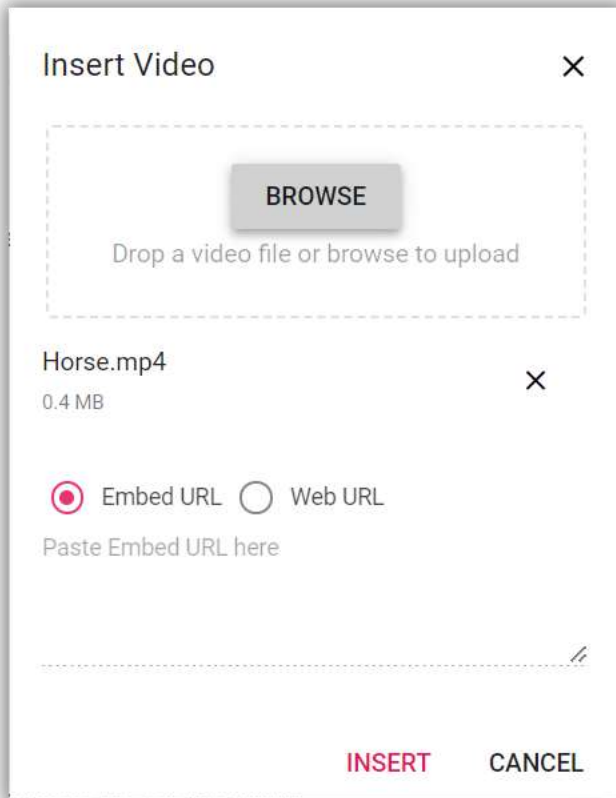
UPDATE

CANCEL

Delete video

To remove a video from the Rich Text Editor content, select the video and click the `videoRemove` button from the quick toolbar. It will delete the video from the Rich Text Editor content as well as from the service location if the [insertVideoSettings.removeUrl](#) is given.

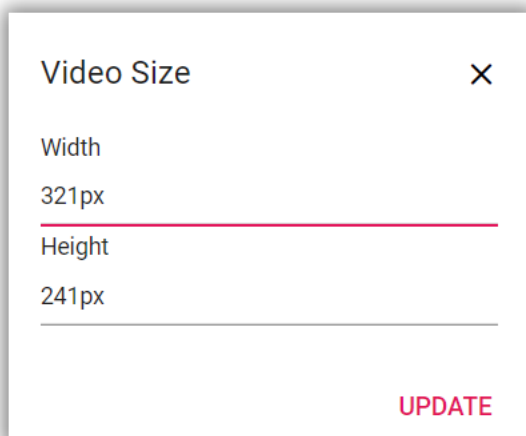
Once you select the video from the local machine, the URL for the video will be generated. You can remove the video from the service location by clicking the cross icon.



Dimension

Set the default width, minWidth, height, and minHeight of the video element, when it is inserted in the Rich Text Editor using the [width](#), [minWidth](#), [height](#), [minHeight](#) properties.

Through the [quickToolbarSettings](#), also you can change the width and height using the **Change Size** button. Once you click on the button, the video size dialog will open as below. In that, specify the width and height of the video in pixels.



Display position

Sets the default display property for the video when it is inserted in the Rich Text Editor using the [insertVideoSettings.layoutOption](#) property. It has two possible options: **Inline** and **Break**. When updating the display positions, it updates the video elements' layout position.

The default **layoutOption** property is set to **Inline**.

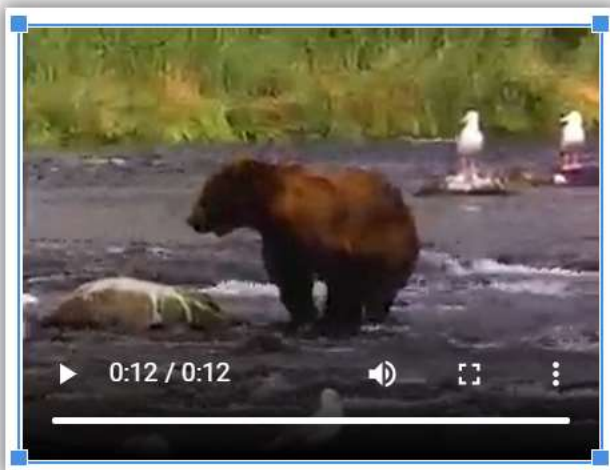
```
`ts
<script>
var defaultRTE = new ej.richtexteditor.RichTextEditor({
insertVideoSettings: {
layoutOption: 'Inline'
}
});
defaultRTE.appendTo('#defaultRTE');
</script>
`
```

Resize video

The Rich Text Editor has built-in video resizing support, which is enabled for the video elements added. The resize points will appear on each corner of the video when focusing so users can easily resize the video using mouse points or thumb through the resize points. Also, the resize calculation will be done based on the aspect ratio.

You can disable the resize action by configuring **false** for the [insertVideoSettings.resize](#) property.

If the [minWidth](#) and [minHeight](#) properties are configured, the video resizing does not shrink below the specified values.



Rename video before inserting

You can use the [insertVideoSettings](#) property, to specify the server handler to upload the selected video. Then by binding the [fileUploadSuccess](#) event, you can receive the modified file name from the server and update it in the Rich Text Editor's insert video dialog.

```
`html
```

```
<div id='defaultRTE'>
```

```
<p>The Rich Text Editor component is WYSIWYG ("what you see is what you get") editor that provides the best user experience to create and update the content.Users can format their content using standard toolbar commands.</p>
```

```
<p><b>Key features:</b></p>
```

```
<ul><li><p>Provides IFRAME and DIV modes</p></li>
```

```
<li><p>Capable of handling markdown editing.</p></li>
```

```
<li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
```

```
<li><p>Provides a fully customizable toolbar.</p></li>
```

```
<li><p>Provides HTML view to edit the source directly for developers.</p></li>
```

```
<li><p>Supports third-party library integration.</p></li>
```

```
<li><p>Allows preview of modified content before saving it.</p></li>
```

```
<li><p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p></li>
```

```
</ul>
```

```
</div>
```

```
`
```

```
`ts
```

```
<script>
```

```
var defaultRTE = new ej.richtexteditor.RichTextEditor({
  toolbarSettings: {
    items: ['Video']
  },
  insertVideoSettings: {
    saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Rename",
    path: "[SERVICEHOSTEDPATH]/Files/"
  },
  fileUploadSuccess: onFileUploadSuccess
});
defaultRTE.appendTo('#defaultRTE');
function onFileUploadSuccess (args) {
```

```

alert("Get the new file name here");
if (args.e.currentTarget.getResponseHeader('name') != null) {
    args.file.name = args.e.currentTarget.getResponseHeader('name');
    var filename = document.querySelectorAll(".e-file-name")[0];
    filename.innerHTML = args.file.name.replace(document.querySelectorAll(".e-file-type")[0].innerHTML,
    "");
    filename.title = args.file.name;
}
}
</script>
`

```

To configure server-side handler, refer to the below code.

```

`c#
int x = 0;
string file;
[AcceptVerbs("Post")]
public void Rename()
{
    try
    {
        var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
        fileName = httpPostedFile.FileName;
        if (httpPostedFile != null)
        {
            var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Files");
            if (!Directory.Exists(fileSave))
            {
                Directory.CreateDirectory(fileSave);
            }
            var fileName = Path.GetFileName(httpPostedFile.FileName);
            var fileSavePath = Path.Combine(fileSave, fileName);
            while (System.IO.File.Exists(fileSavePath))
            {

```

```

fileName = "rteFiles" + x + "-" + fileName;
fileSavePath = Path.Combine(fileSave, fileName);
x++;
}
if (!System.IO.File.Exists(fileSavePath))
{
    httpPostedFile.SaveAs(fileSavePath);
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
    Response.Clear();
    Response.Headers.Add("name", fileName);
    Response.ContentType = "application/json; charset=utf-8";
    Response.StatusDescription = "File uploaded succesfully";
    Response.End();
}
}
}
catch (Exception e)
{
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
    Response.Clear();
    Response.ContentType = "application/json; charset=utf-8";
    Response.StatusCode = 204;
    Response.Status = "204 No Content";
    Response.StatusDescription = e.Message;
    Response.End();
}
}
,

```

Upload video with authentication

You can add additional data with the video uploaded from the Rich Text Editor on the client side, which can even be received on the server side by using the [fileUploading](#) event and its `customFormData` argument, you can pass parameters to the controller action. On the server side, you can fetch the custom headers by accessing the form collection from the current request, which retrieves the values sent using the POST method.

By default, it doesn't support the `UseDefaultCredentials` property, you can manually append the default credentials with the upload request.

```
`ts
<script>
var defaultRTE = new ej.richtexteditor.RichTextEditor({
toolbarSettings: {
items: ['Video']
},
insertVideoSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/SaveFiles",
path: "[SERVICEHOSTEDPATH]/Files/"
},
fileUploading: onFileUpload
});
defaultRTE.appendTo('#defaultRTE');
function onFileUpload (args) {
var accessToken = "Authorization_token";
// adding custom Form Data
args.customFormData = [{ 'Authorization': accessToken }];
}
</script>
`c#
public void SaveFiles(IList<IFormFile> UploadFiles)
{
string currentPath = Request.Form["Authorization"].ToString();
}
```

See Also

- [How to edit the quick toolbar settings](#)
- [How to use the link editing option in the toolbar items](#)

Link in EJ2 JavaScript Rich text editor control

A hyperlink can be insert into the editor for quick access to the related information. The hyperlink itself can be a text or an image.

Insert link

Point the cursor anywhere within the editor where you would like to insert the link. It is also possible to select a text or an image within the editor and can be converted to the hyperlink. Click the Insert HyperLink tool on the toolbar. The Insert Link Dialog will be open. The dialog has the following options.

Rich Text Editor features are segregated into individual feature-wise modules. To use image and link tool, inject link module using the `RichTextEditor.Inject(Link)`.

- | Options | Description |
|-------------------------|---|
| Web Address | Type or paste the destination for the link you are creating |
| Display Text | Type or edit the required text that you want to display text for the link |
| Tooltip | To display additional helpful information when you place the pointer on the hyperlink, type the required text in the "Tooltip" field. |
| Open Link in New Window | Specify whether, the given link will be open in new window or not |

The Rich Text Editor link tool validates the URLs, as you type them in Web Address. URLs considered invalid will be highlighted with red color by clicking the insert button in the `Insert Link` dialog.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
```

```

* Rich Text Editor default sample
*/
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({toolbarSettings: {
    items: ['CreateLink', 'RemoveLink']} });
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
      <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>

```



```

        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remove Link

If you want to remove a hyperlink from a text or image, select the text or image with the hyperlink and click "Remove Hyperlink" tool from the toolbar. It will keep the text or image.

Auto-link

When you type URL, and Enter key to the Rich Text Editor, the typed URL will be automatically changed into the hyperlink.

Manipulation

Add the custom tools on the selected link inside the Rich Text Editor through the quick toolbar.



The quick toolbar for the link has the following options.

Tools	Description
Open	The given link page, will be open in new window.
Edit Link	Used to edit the link in the Rich Text Editor content.
Remove Link	Used to remove link from the content of Rich Text Editor.
Custom Tool	Used to add the custom options in the quick toolbar.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({toolbarSettings: {
    items: ['CreateLink', 'RemoveLink']} });
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
      get") editor useful to create and edit content, and return the valid <a
      href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
      <a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
      the content</p>
      <p><b>Key features:</b></p>
```

```

        <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
        <li><p>Capable of handling markdown editing.</p></li>
        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
        </ul>
    </div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to edit the quick toolbar settings](#)
- [How to insert image link editing option in the toolbar items](#)

Table in EJ2 JavaScript Rich text editor control

Rich Text Editor allows to insert table of content in edit panel and provide options to add, edit, and remove the table as well as perform other table related action. For inserting the table to the Rich Text Editor, the following list of options have been provided in the [tableSettings](#)

Options	Description	Default Value
minWidth	Sets the default minWidth of the table.	0
maxWidth	Sets the default maxWidth of the table.	null
resize	Enable resize feature in table.	true

| styles | This is an array of key value pair, on each pair, key should be name of styling and value is class name. this list will be shown on quick toolbar options to change the styles of table on designing like dashed, double bordered. | [TableStyleItems](#) |

| width | Sets the default width of the table. | 100% |

Rich Text Editor features are segregated into individual feature-wise modules. To use table tool, inject image module using the `RichTextEditor.Inject(Table)`.

Insert table

Using the `table` toolbar option, select a number of rows and columns to be inserted over the table grid and insert table into Rich Text Editor content using the mouse.

Tables can also be inserted through the `Insert Table` option in the pop-up where the number of rows and columns can be provided manually, and this is the default way in devices.

In the following sample, the table has been injected from table module.

INDEX.TS

```
import { RichTextEditor, Toolbar, Image, Link, HtmlEditor, QuickToolbar,
NodeSelection, Table } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Image, Link, HtmlEditor, QuickToolbar, Table
);
let defaultRTE: RichTextEditor = new RichTextEditor({
    toolbarSettings: {
        items: ['CreateTable']}
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Quick Toolbar

Quick toolbar is opened by clicking the table. It has different sets of commands to be performed on the table which increases the feasibility to edit the table easily.

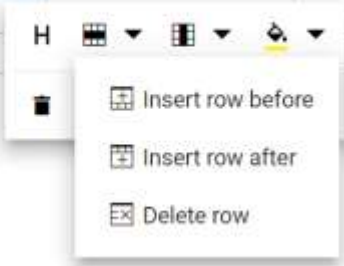
To use quick toolbar, inject the quick toolbar module using the `RichTextEditor.Inject(QuickToolbar)`.

Table Header

Table Header command is available with quick toolbar option through which the header row can be added or removed from the inserted table. The following image illustrates the table header.

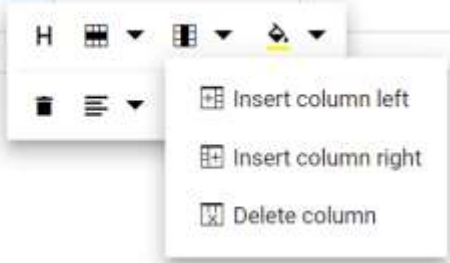
Insert Rows

Rows can be inserted above or below the required table cell through the quick toolbar. Also, focused row can be deleted. The following screenshot shows the available options of the row item.



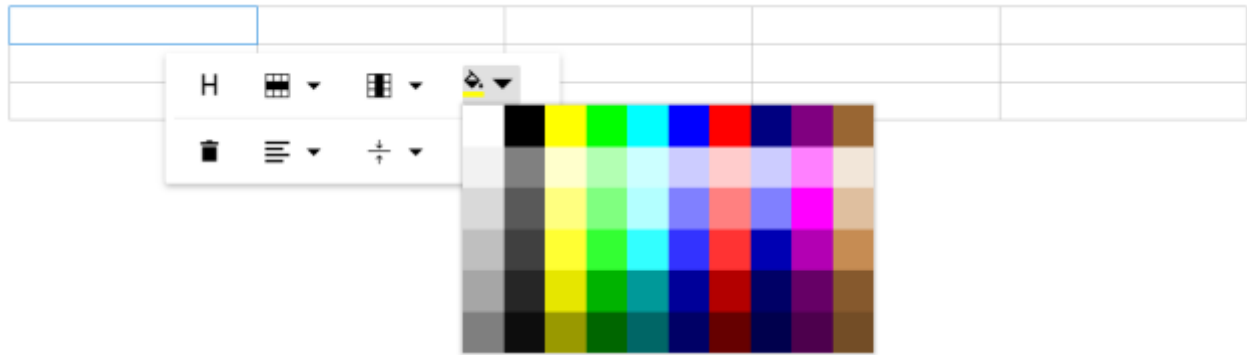
Insert Columns

Columns can be inserted to the left or right side of the required table cell through the quick toolbar. Also, the focused column can be deleted. The following screenshot shows the available options of the column item.



Set Color

The background color can be set for each table cell through the **background color** command available with quick toolbar.



Delete Table

Using the delete item in the quick toolbar, users can delete the entire table.

Vertical Align

Text inside the table can be aligned to top, middle, or bottom using the [tableCellVerticalAlign](#) tool of the quick toolbar.



Horizontal Align

Text inside the table can be aligned left, right, or center using the [tableCellHorizontalAlign](#) tool of the quick toolbar.

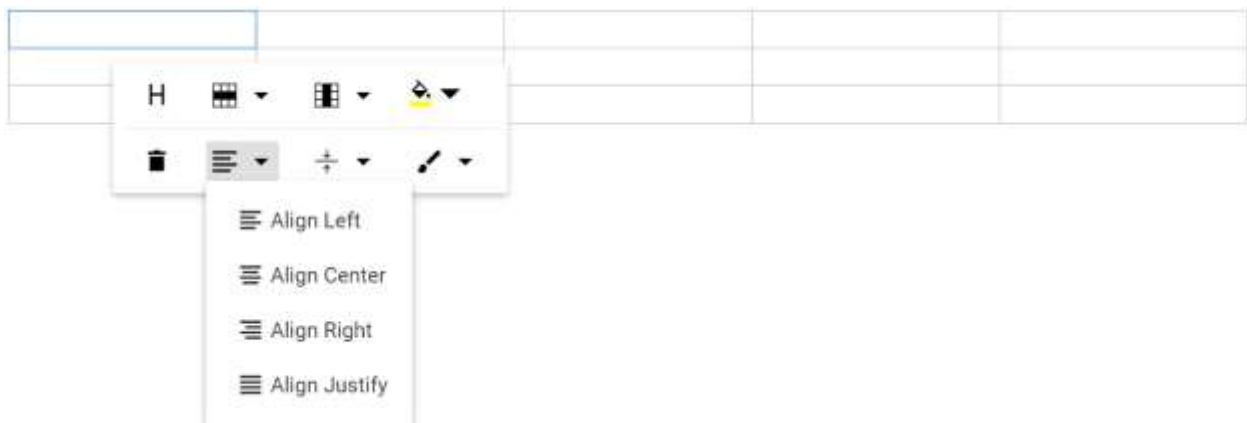


Table Styles

Table styles provided for class name should be appended to a table element. It helps to design the table in specific CSS styles when inserting in the editor.

By Default, provides Dashed border and Alternate rows.

Dashed border: Applies the dashed border to the table.

Alternate border: Applies the alternative background to the table.

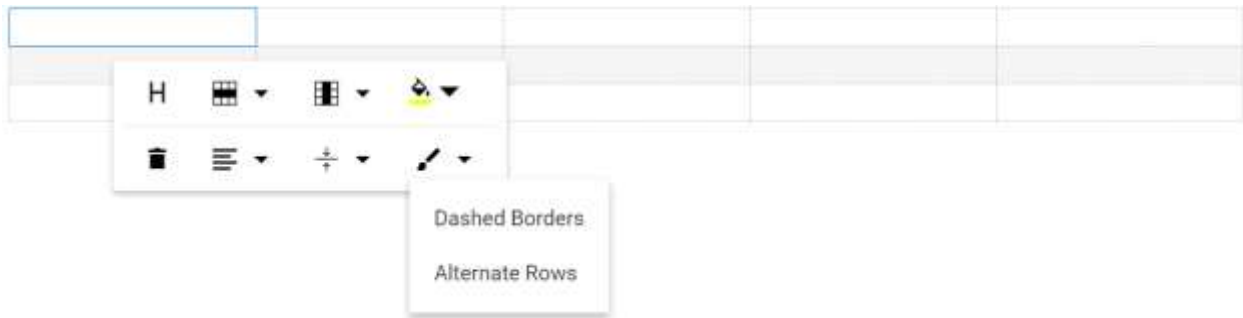


Table Properties

Sets the default width of the table when it is inserted in the Rich Text Editor using the width of [tableSettings](#).

Using the quick toolbar, users can change the width, cell padding, and cell spacing in the selected table using the properties option.

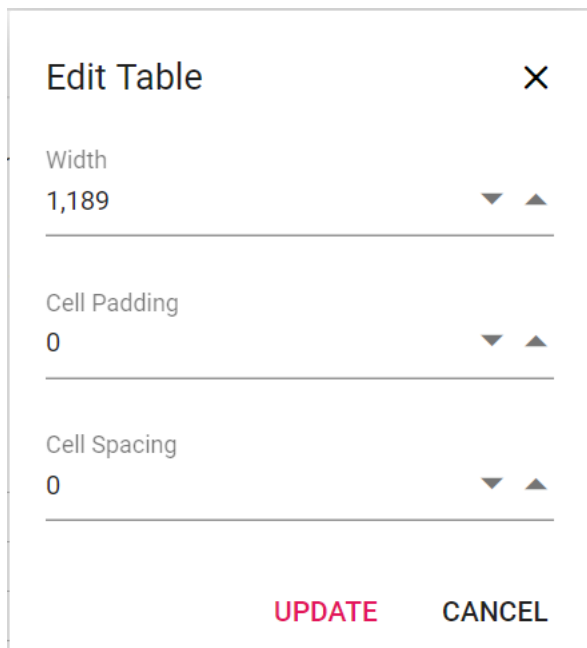


Table cell merge and split

The Rich Text Editor allows users to change the appearance of the tables by splitting or merging the table cells.

TableCell item should be configured in the Table [quickToolbarSettings](#) Property to show the merge/split icons while selecting the table cells

Table cell merge

The table cell merge feature allows you to merge two or more row and column cells into a single cell with its contents.

The following image explains the table merge action.

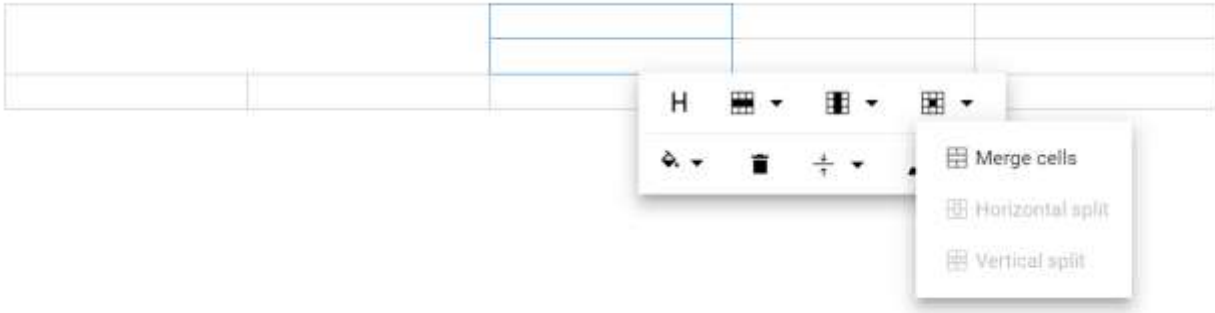


Table cell split

The table cell split feature allows you to a selected cell can be split both horizontally and vertically.

The following image explains the table split action.



INDEX.TS

```
import { RichTextEditor, Toolbar, Image, Link, HtmlEditor, QuickToolbar,
NodeSelection, Table } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Image, Link, HtmlEditor, QuickToolbar, Table
);
let defaultRTE: RichTextEditor = new RichTextEditor({
    toolbarSettings: {
        items: ['CreateTable']},
    quickToolbarSettings: {
        table: ['TableHeader', 'TableRows', 'TableColumns', 'TableCell',
'-'],
        'BackgroundColor', 'TableRemove', 'TableCellVerticalAlign',
'Styles']
    },
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you get") editor useful to create and edit content, and return the valid <a href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or <a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62; modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Emoji Picker in EJ2 JavaScript RichTextEditor control

An emoji picker is a tool that allows users to add emojis or emoticons to their text easily. Typically, it is a small window or panel that displays a variety of emojis arranged in different categories, such as smileys, animals, food, and so on. Users can select the desired emoji by clicking on it or typing its name in a search bar.

Enabling the toolbar option and custom emojis.

Add the **EmojiPicker** tool to the toolbar of the RichTextEditor by utilizing the **toolbarSettings items** property.

By default, a predefined set of emojis is configured. However, you can customize these icons according to your needs. To achieve this, utilize the [emojiPickerSettings](#) property.

```
`ts
```

```
<script>
```

```
var defaultRTE: RichTextEditor = new ej.richtexteditor.RichTextEditor({
```

```
toolbarSettings: {
```

```
items: ['EmojiPicker']
```

```
},
```

```
emojiPickerSettings: {
```

```
iconsSet: [{name: 'Smilies & People', code: '1F600', iconCss: 'e-emoji',
```

```
icons: [{ code: '1F600', desc: 'Grinning face' },
```

```
{ code: '1F603', desc: 'Grinning face with big eyes' },
```

```
{ code: '1F604', desc: 'Grinning face with smiling eyes' },
```

```
{ code: '1F606', desc: 'Grinning squinting face' },
```

```
{ code: '1F605', desc: 'Grinning face with sweat' },
```

```
{ code: '1F602', desc: 'Face with tears of joy' },
```

```
{ code: '1F923', desc: 'Rolling on the floor laughing' },
```

```
{ code: '1F60A', desc: 'Smiling face with smiling eyes' }]
```

```
}, {
```

```
name: 'Animals & Nature', code: '1F435', iconCss: 'e-animals',
```

```
icons: [{ code: '1F436', desc: 'Dog face' },
```

```
{ code: '1F431', desc: 'Cat face' },
```

```
{ code: '1F42D', desc: 'Mouse face' },
{ code: '1F439', desc: 'Hamster face' },
{ code: '1F430', desc: 'Rabbit face' },
{ code: '1F98A', desc: 'Fox face' }]
}, {
name: 'Food & Drink', code: '1F347', iconCss: 'e-food-and-drinks',
icons: [{ code: '1F34E', desc: 'Red apple' },
{ code: '1F34C', desc: 'Banana' },
{ code: '1F347', desc: 'Grapes' },
{ code: '1F353', desc: 'Strawberry' },
{ code: '1F35E', desc: 'Bread' },
{ code: '1F950', desc: 'Croissant' },
{ code: '1F955', desc: 'Carrot' },
{ code: '1F354', desc: 'Hamburger' }]
}, {
name: 'Activities', code: '1F383', iconCss: 'e-activities',
icons: [{ code: '26BD', desc: 'Soccer ball' },
{ code: '1F3C0', desc: 'Basketball' },
{ code: '1F3C8', desc: 'American football' },
{ code: '26BE', desc: 'Baseball' },
{ code: '1F3BE', desc: 'Tennis' },
{ code: '1F3D0', desc: 'Volleyball' },
{ code: '1F3C9', desc: 'Rugby football' }]
}, {
name: 'Travel & Places', code: '1F30D', iconCss: 'e-travel-and-places',
icons: [{ code: '2708', desc: 'Airplane' },
{ code: '1F697', desc: 'Automobile' },
{ code: '1F695', desc: 'Taxi' },
{ code: '1F6B2', desc: 'Bicycle' },
{ code: '1F68C', desc: 'Bus' }]
}, {
name: 'Objects', code: '1F507', iconCss: 'e-objects', icons: [{ code: '1F4A1', desc: 'Light bulb' },
{ code: '1F526', desc: 'Flashlight' },
```

```

{ code: '1F4BB', desc: 'Laptop computer' },
{ code: '1F5A5', desc: 'Desktop computer' },
{ code: '1F5A8', desc: 'Printer' },
{ code: '1F4F7', desc: 'Camera' },
{ code: '1F4F8', desc: 'Camera with flash' },
{ code: '1F4FD', desc: 'Film projector' }}
}, {
name: 'Symbols', code: '1F3E7', iconCss: 'e-symbols', icons: [{ code: '274C', desc: 'Cross mark' },
{ code: '2714', desc: 'Check mark' },
{ code: '26A0', desc: 'Warning sign' },
{ code: '1F6AB', desc: 'Prohibited' },
{ code: '2139', desc: 'Information' },
{ code: '267B', desc: 'Recycling symbol' },
{ code: '1F6AD', desc: 'No smoking' }]
}]
}
});
defaultRTE.appendTo('#defaultRTE');
</script>
`

```

Additionally, you have the option to customize the icons of toolbar items using the [iconCss](#) and [code](#) properties. The `iconCSS` property allows you to define a custom CSS class for the toolbar item icon, while the `code` property enables you to specify the Unicode character code for the icon.

When both `iconCSS` and `code` properties are provided, the `iconCSS` property takes precedence in determining the appearance of the toolbar item icon.

Additionally, you have the option to enhance the user experience by implementing a filtering feature for efficiently managing a large dataset of emojis. By setting the [showSearchBox](#) property to true (which is the default value), users will be able to utilize a search box to filter the displayed emojis according to their preferences.

The following code example shows how to add the emoji picker tool in the RichTextEditor.

INDEX.JS

```

/**
 * Rich Text Editor - Emoji picker sample
 */
var defaultRTE = new ej.richtexteditor.RichTextEditor({
  toolbarSettings: {

```

```

        items: ['Bold', 'Italic', 'Underline', '|', 'Formats', 'Alignments',
'OrderedList',
        'UnorderedList', '|', 'CreateLink', 'Image', '|', 'SourceCode',
'EmojiPicker',
        '|', 'Undo', 'Redo']
    },
});
defaultRTE.appendTo('#emojiPickerRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id='emojiPickerRTE'>
      <p>An emoji picker in a Rich Text Editor is a tool that allows
users to easily add emojis or emoticons to their text.</p>
      <p>Typically, it is a small window or panel that displays a
variety of emojis, arranged in different categories, such as smileys,
animals, food, and so on. Users can select the desired emoji by clicking on
it or by typing its name in a search bar.</p>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Using the shortcut key to open the emoji picker

Quickly access the emoji picker by simply pressing the colon (:) key while typing a word prefix in an editor, allowing for instant emoji selection and display. Moreover, continue typing in the editor after the colon (:) to filter and refine your search for the desired emojis.

![JavaScript Rich Text Editor Emoji Picker shortcut](../images/javascript-richtexteditor-emoji-picker-shortcut.png)

Navigating and selecting emojis using the keyboard

The emoji picker popup offers keyboard navigation options, allowing you to move the emoji focus from one emoji to another. The following keys are used for navigation:

Arrow keys: Use the arrow keys (up, down, left, right) to move the emoji focus in the corresponding direction.

Enter: Press Enter key to select the currently focused emoji.

Escape: Press Escape to close the emoji picker popup without selecting an emoji.

Markdown in EJ2 JavaScript Rich text editor control

When you format the word in Markdown format, you should add Markdown syntax to the word to indicate the words and phrases that looks different from each other.

Rich Text Editor supports markdown editing when the [editorMode](#) set as **markdown** and using both *keyboard interaction* and *toolbar action*, you can apply the formatting to text.

To create Rich Text Editor with Markdown editing feature, inject the [MarkdownEditor](#) module to the Rich Text Editor using the `RichTextEditor.Inject(MarkdownEditor)` method.

Supported Commands

The Javascript Markdown editor supports the following commands to format the markdown content:

Commands	Syntax	Description
Bold	Sample content for bold text .	For bold, add or to front and back of the text. For order list, precede each line with a number.
Italic	Sample content for <i>Italic text</i> .	For Italic, add * or _ to front and back of the text.
Bold and Italics	Sample content for <i>bold and Italic text</i> .	For bold and Italics, add * to the front and back of the text.
Heading 1	# Heading 1 content	For heading 1, add # to start of the line.
Heading 2	## Heading 2 content	For heading 2, add ## to start of the line.
Heading 3	### Heading 3 content	For heading 3, add ### to start of the line.
Heading 4	#### Heading 4 content	For heading 4, add #### to start of the line.
Heading 5	##### Heading 5 content	For heading 5, add ##### to start of the line.

| Heading 6 | ##### Heading 6 content | For heading 6, add ##### to start of the line. |

| Line Break | First line
Second line | For line break, press enter two times (or) add
 in between the first and the second line. |

| Blockquotes | > Blockquotes text | For blockquotes, add > to start of the line. |

| Strike Through | Sample content for ~~strike through text~~. | For strike through, add ~ to front and back of the text. |

| Code (Single line) | \Single line code\ | For single line code, add ` to front and back of the text. |

| Code block (Multi Line) | \\
Multi line code text
Multi line code text
\\ | For multiple line code, add \\ in the new line before and after the content. |

| Subscript | _{Subscript text} | For subscript, add _{to the front and} to the back of the text. |

| Superscript | ^{Superscript text} | For superscript, add ^{to the front and} to the back of the text. |

| Ordered List | 1. First
1. Second | For ordered list, preceding one or more lines of text with 1. |

| Unordered List | *First*
second | For unordered list, preceding one or more lines of text with *. |

| Links | **Link text without title text**
`Link text`
Link text with title text
[Link text](URL, "title text") | Create an inline link by wrapping link text in brackets [], and then wrapping the URL as first parameter and title as second parameter in the parentheses ().
Note: The title text is optional, if needed it can be given manually. |

| Table | | Heading 1 | Heading 2 |
|-----|-----|
| Col A1 | Col A2 |
| Col B1 | Col B2 | | Create a table using the pipes and underscores as given in the syntax to create 2 x 2 table. |

| Horizontal Line | *(three asterix in new line)*
(or)
(three underscores in new line) | **For horizontal line, add or** to the start of the new line. |

| Image | | Create an image by wrapping the image source in parentheses (). |

| Image with alternate text | ![alternate text](URL path) | Create an image with alternate text by wrapping an alternative text in brackets [], and then link of the image source in parentheses ().
Note: When inserting the image using toolbar, the alternate text cannot be provided that needs to be given manually. |

| Escape tick marks supported | Sample text content with **bold and not bold text can be in the same line.** | In the syntax, the whole content is made as bold where the content not bold can be made as normal text by adding the bold syntax to the start and end of the respective text. Likewise you can do the same for various inline commands. |

| Escape Character | \ (any syntax) | Escape any markdown syntax by prefix \ to the syntax.
Example:
Bold text|

| HTML Entities | Copyright: © - ©
Trade mark: ™ - ™
Registered: ® - ®
Ampersand: & - &
Less than: <

<
Greater than: > - > | For HTML entities, add & and ; to the front and back of the respective entities. |

The above listed commands alone are supported in Syncfusion Markdown editor. For other unsupported commands, you can achieve using the HTML tags in Markdown editor. The foot notes, definitions, math, and check list markdown syntax are also not supported.

Markdown to HTML

The Rich Text Editor allows you to preview markdown changes immediately using preview. In this sample, the third-party library [Marked](#) is used to convert markdown into HTML content.

This sample demonstrates how to preview markdown changes in Rich Text Editor. Type or edit the display text, and apply format to view the preview of markdown. The [actionComplete](#) event can be used to convert Markdown to HTML.

This sample demonstrates how to preview markdown changes in Rich Text Editor. Type or edit the display text and apply format to view the preview of markdown.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Link, Image, MarkdownEditor, Toolbar, QuickToolbar } from '@syncfusion/ej2-richtexteditor';
import { createElement, KeyboardEventArgs, isNullOrUndefined, addClass, removeClass, Browser } from '@syncfusion/ej2-base';
RichTextEditor.Inject(Link, Image, MarkdownEditor, Toolbar, QuickToolbar);
let textArea: HTMLTextAreaElement;
let mdsource: HTMLElement;
let mdSplit: HTMLElement;
let htmlPreview: HTMLElement;
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340, editorMode: 'Markdown',
    toolbarSettings: {
        items: ['Bold', 'Italic', 'StrikeThrough', '|', 'Formats', 'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', '|',
            { tooltipText: 'Preview', template: '<button id="preview-code" class="e-tbar-btn e-control e-btn e-icon-btn"> <span class="e-btn-icon e-md-preview e-icons"></span></button>', '|', 'Undo', 'Redo']
        },
        created: () => {
            textArea = defaultRTE.contentModule.getEditPanel() as HTMLTextAreaElement;
            textArea.addEventListener('keyup', (e: KeyboardEventArgs) => {
                markDownConversion();
            });
            let rteObj: RichTextEditor = defaultRTE;
            mdsource = document.getElementById('preview-code');
            mdsource.addEventListener('click', (e: MouseEvent) => {
                fullPreview({ mode: true, type: 'preview' });
                if ((e.currentTarget as HTMLElement).classList.contains('e-active')) {
                    defaultRTE.disableToolbarItem(['Bold', 'Italic', 'StrikeThrough', '|', 'Formats', 'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', 'Undo', 'Redo']);
                }
            });
        }
    }
});
```

```

        } else {
            defaultRTE.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
            'Formats', 'OrderedList', 'UnorderedList', '|',
            'CreateLink', 'Image', 'Undo', 'Redo']);
        }
    });
},
});
defaultRTE.appendTo('#defaultRTE');
function markdownConversion(): void {
    if (mdsource.classList.contains('e-active')) {
        let id: string = defaultRTE.getID() + 'html-preview';
        let htmlPreview: HTMLElement = defaultRTE.element.querySelector('#' + id);
        let rteElement = defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement;
        let rteValue = rteElement.value;
        htmlPreview.innerHTML =
marked((defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement).value);
    }
}
function fullPreview(e: { [key: string]: string | boolean }): void {
    let id: string = defaultRTE.getID() + 'html-preview';
    htmlPreview = defaultRTE.element.querySelector('#' + id);
    if (mdsource.classList.contains('e-active')) {
        mdsource.classList.remove('e-active');
        mdsource.parentElement.title = 'Preview';
        textArea.style.display = 'block';
        textArea.style.width = '100%';
        htmlPreview.style.display = 'none';
    } else {
        mdsource.classList.add('e-active');
        if (!htmlPreview) {
            htmlPreview = createElement('div', { className: 'e-content' });
            htmlPreview.id = id;
            textArea.parentNode.appendChild(htmlPreview);
        }
        if (e.type === 'preview') {
            textArea.style.display = 'none'; htmlPreview.classList.add('e-
pre-source');
        } else {
            htmlPreview.classList.remove('e-pre-source');
            textArea.style.width = '50%';
        }
        htmlPreview.style.display = 'block';
        markdownConversion();
        mdsource.parentElement.title = 'Code View';
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Rich Text Editor</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
        <div id="defaultRTE">
In Rich Text Editor , you click the toolbar buttons to format the words and
the changes are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to add Markdown syntax to the word to indicate which words
and phrases should look different from each other
Rich Text Editor supports markdown editing when the editorMode set as
**markdown** and using both *keyboard interaction* and *toolbar action*, you
can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation,
[sample link] (https://ej2.syncfusion.com/home/).
The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content
        </div>
    </div>
    <style>
        .e-richtexteditor textarea.e-content {
            float: left;
            border-right: 1px solid rgba(0, 0, 0, 0.12);
        }
        .e-richtexteditor .e-rte-content .e-content {
            min-height: 150px;
        }
    </style>

```

```

.e-richtexteditor .e-rte-content {
    overflow: hidden;
}

.e-icon-btn.e-active .e-md-preview::before {
    content: '\e350';
}

.e-icon-btn .e-md-preview::before {
    content: '\e345';
}

.e-rte-content .e-content {
    float: right;
    width: 50%;
    overflow: auto;
    height: inherit;
    padding: 8px;
    height: 100%;
}

.e-rte-content .e-content.e-pre-source {
    width: 100%;
}

.highcontrast .e-richtexteditor textarea.e-content {
    border-right: 1px solid #fff;
}

.sb-header {
    z-index: 100;
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Table

Rich Text Editor allows to insert Markdown table in edit panel with 2 X 2 rows and columns along with the heading.

To use table tool, add the **CreateTable** item in toolbar items.

Insert table

To insert the table in Rich Text Editor, click the **table** toolbar option to insert the table into Rich Text Editor content and this is the default way in all the devices.

Please refer the below sample and code snippets to add the table in Markdown editor

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Link, Image, MarkdownEditor, Toolbar, QuickToolbar
} from '@syncfusion/ej2-richtexteditor';
import { createElement, KeyboardEventArgs, isNullOrUndefined, addClass,
removeClass, Browser } from '@syncfusion/ej2-base';
RichTextEditor.Inject(Link, Image, MarkdownEditor, Toolbar, QuickToolbar);
let textArea: HTMLTextAreaElement;
let mdsource: HTMLInputElement;
let mdSplit: HTMLInputElement;
let htmlPreview: HTMLInputElement;
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340, editorMode: 'Markdown',
    toolbarSettings: {
        items: ['Bold', 'Italic', 'StrikeThrough', '|', 'Formats',
'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', 'CreateTable',
'|',
            { tooltipText: 'Preview', template: '<button id="preview-code"
class="e-tbar-btn e-control e-btn e-icon-btn"> +
                '<span class="e-btn-icon e-md-preview e-
icons"></span></button>' }, '|', 'Undo', 'Redo']
        },
    created: () => {
        textArea = defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement;
        textArea.addEventListener('keyup', (e: KeyboardEventArgs) => {
markDownConversion(); });
        let rteObj: RichTextEditor = defaultRTE;
        mdsource = document.getElementById('preview-code');
        mdsource.addEventListener('click', (e: MouseEvent) => {
            fullPreview({ mode: true, type: 'preview' });
            if ((e.currentTarget as HTMLInputElement).classList.contains('e-
active')) {
                defaultRTE.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
                    'Formats', 'OrderedList', 'UnorderedList', '|',
                    'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
            } else {
                defaultRTE.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
                    'Formats', 'OrderedList', 'UnorderedList', '|',
                    'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
            }
        });
    },
});
defaultRTE.appendTo('#defaultRTE');
function markDownConversion(): void {
    if (mdsource.classList.contains('e-active')) {
        let id: string = defaultRTE.getID() + 'html-preview';
        let htmlPreview: HTMLInputElement = defaultRTE.element.querySelector('#'
+ id);
        let rteElement = defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement;
        let rteValue = rteElement.value;

```

```

        htmlPreview.innerHTML =
marked((defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement).value);
    }
}
function fullPreview(e: { [key: string]: string | boolean }): void {
    let id: string = defaultRTE.getID() + 'html-preview';
    htmlPreview = defaultRTE.element.querySelector('#' + id);
    if (mdsource.classList.contains('e-active')) {
        mdsource.classList.remove('e-active');
        mdsource.parentElement.title = 'Preview';
        textArea.style.display = 'block';
        textArea.style.width = '100%';
        htmlPreview.style.display = 'none';
    } else {
        mdsource.classList.add('e-active');
        if (!htmlPreview) {
            htmlPreview = createElement('div', { className: 'e-content' });
            htmlPreview.id = id;
            textArea.parentNode.appendChild(htmlPreview);
        }
        if (e.type === 'preview') {
            textArea.style.display = 'none'; htmlPreview.classList.add('e-
pre-source');
        } else {
            htmlPreview.classList.remove('e-pre-source');
            textArea.style.width = '50%';
        }
        htmlPreview.style.display = 'block';
        markDownConversion();
        mdsource.parentElement.title = 'Code View';
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
        <div id="defaultRTE">
In Rich Text Editor , you click the toolbar buttons to format the words and
the changes are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to add Markdown syntax to the word to indicate which words
and phrases should look different from each other
Rich Text Editor supports markdown editing when the editorMode set as
**markdown** and using both *keyboard interaction* and *toolbar action*, you
can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation,
[sample link] (https://ej2.syncfusion.com/home/).
The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content
        </div>
    </div>
    <style>
        .e-richtexteditor textarea.e-content {
            float: left;
            border-right: 1px solid rgba(0, 0, 0, 0.12);
        }
        .e-richtexteditor .e-rte-content .e-content {
            min-height: 150px;
        }

        .e-richtexteditor .e-rte-content {
            overflow: hidden;
        }

        .e-icon-btn.e-active .e-md-preview::before {
            content: '\e350';
        }

        .e-icon-btn .e-md-preview::before {
            content: '\e345';
        }

        .e-rte-content .e-content {
            float: right;
            width: 50%;
            overflow: auto;

```

```

        height: inherit;
        padding: 8px;
        height: 100%;
    }

    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }

    .highcontrast .e-richtexteditor textarea.e-content {
        border-right: 1px solid #fff;
    }

    .sb-header {
        z-index: 100;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing table constants

The Markdown table constants can be changed for the table heading and the column name.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Link, Image, MarkdownEditor, Toolbar, QuickToolbar } from '@syncfusion/ej2-richtexteditor';
import { createElement, KeyboardEventArgs, isNullOrUndefined, addClass, removeClass, Browser } from '@syncfusion/ej2-base';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
    'en-US': {
        'richtexteditor': {
            'TableHeadingText': 'Header',
            'TableColText': 'Cell'
        }
    }
});
RichTextEditor.Inject(Link, Image, MarkdownEditor, Toolbar, QuickToolbar);
let textArea: HTMLTextAreaElement;
let mdsource: HTMLInputElement;
let mdSplit: HTMLInputElement;
let htmlPreview: HTMLInputElement;
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340, editorMode: 'Markdown',
    toolbarSettings: {

```



```

        items: ['Bold', 'Italic', 'StrikeThrough', '|', 'Formats',
'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', 'CreateTable',
'|',
        { tooltipText: 'Preview', template: '<button id="preview-code"
class="e-tbar-btn e-control e-btn e-icon-btn">' +
        '<span class="e-btn-icon e-md-preview e-
icons"></span></button>' }, '|', 'Undo', 'Redo']
    },
    created: () => {
        textArea = defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement;
        textArea.addEventListener('keyup', (e: KeyboardEventArgs) => {
markDownConversion(); });
        let rteObj: RichTextEditor = defaultRTE;
        mdsource = document.getElementById('preview-code');
        mdsource.addEventListener('click', (e: MouseEvent) => {
            fullPreview({ mode: true, type: 'preview' });
            if ((e.currentTarget as HTMLElement).classList.contains('e-
active')) {
                defaultRTE.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
                'Formats', 'OrderedList', 'UnorderedList', '|',
                'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
            } else {
                defaultRTE.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
                'Formats', 'OrderedList', 'UnorderedList', '|',
                'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
            }
        });
    },
});
defaultRTE.appendTo('#defaultRTE');
function markDownConversion(): void {
    if (mdsource.classList.contains('e-active')) {
        let id: string = defaultRTE.getID() + 'html-preview';
        let htmlPreview: HTMLElement = defaultRTE.element.querySelector('#'
+ id);
        let rteElement = defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement;
        let rteValue = rteElement.value;
        htmlPreview.innerHTML =
marked((defaultRTE.contentModule.getEditPanel() as
HTMLTextAreaElement).value);
    }
}
function fullPreview(e: { [key: string]: string | boolean }): void {
    let id: string = defaultRTE.getID() + 'html-preview';
    htmlPreview = defaultRTE.element.querySelector('#' + id);
    if (mdsource.classList.contains('e-active')) {
        mdsource.classList.remove('e-active');
        mdsource.parentElement.title = 'Preview';
        textArea.style.display = 'block';
        textArea.style.width = '100%';
        htmlPreview.style.display = 'none';
    } else {
        mdsource.classList.add('e-active');

```

```

    if (!htmlPreview) {
        htmlPreview = createElement('div', { className: 'e-content' });
        htmlPreview.id = id;
        textArea.parentNode.appendChild(htmlPreview);
    }
    if (e.type === 'preview') {
        textArea.style.display = 'none'; htmlPreview.classList.add('e-
pre-source');
    } else {
        htmlPreview.classList.remove('e-pre-source');
        textArea.style.width = '50%';
    }
    htmlPreview.style.display = 'block';
    markDownConversion();
    mdsource.parentElement.title = 'Code View';
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>

```

```
<div id="defaultRTE">
```

In Rich Text Editor , you click the toolbar buttons to format the words and the changes are visible immediately.

Markdown **is** not like that. When you format the word **in** Markdown format, you need to **add** Markdown syntax to the word to indicate which words and phrases should look different **from** each other

Rich Text Editor supports markdown editing when the editorMode **set as** ****markdown**** and **using both** ***keyboard interaction*** and ***toolbar action***, you can apply the formatting to text.

We can **add** our own custom formation syntax **for** the Markdown formation, [sample link] (<https://ej2.syncfusion.com/home/>).

The third-party library `Marked` **is** used **in this** sample to convert markdown **into** HTML content

```
</div>
</div>
<style>
    .e-richtexteditor textarea.e-content {
        float: left;
        border-right: 1px solid rgba(0, 0, 0, 0.12);
    }
    .e-richtexteditor .e-rte-content .e-content {
        min-height: 150px;
    }

    .e-richtexteditor .e-rte-content {
        overflow: hidden;
    }

    .e-icon-btn.e-active .e-md-preview::before {
        content: '\e350';
    }

    .e-icon-btn .e-md-preview::before {
        content: '\e345';
    }

    .e-rte-content .e-content {
        float: right;
        width: 50%;
        overflow: auto;
        height: inherit;
        padding: 8px;
        height: 100%;
    }

    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }

    .highcontrast .e-richtexteditor textarea.e-content {
        border-right: 1px solid #fff;
    }

    .sb-header {
        z-index: 100;
    }
</style>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Custom format

The Rich Text Editor allows you to customize the markdown syntax by overriding its default syntax. Configure the customized markdown syntax using the [formatter](#) property.

This sample demonstrates how to customize tags of markdown formatting.

For example, apply **+** to Unordered list, apply **1., 2., 3.** to Ordered list, for bold, **__**, and for italic **_**.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, MarkdownFormatter,
MarkdownEditor, QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, MarkdownEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
  height: 340,
  toolbarSettings: {
    items: ['Bold', 'Italic', 'StrikeThrough', '|',
      'Formats', 'OrderedList', 'UnorderedList', '|',
      'CreateLink', 'Image', '|', 'Undo', 'Redo']
  },
  editorMode: 'Markdown',
  formatter: new MarkdownFormatter({
    listTags: { 'OL': '1., 2., 3.', 'UL': '+ ' },
    formatTags: {
      'Blockquote': '> '
    },
    selectionTags: { 'Bold': '__', 'Italic': '_' }
  })
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
        <div id="defaultRTE">
In Rich Text Editor , you click the toolbar buttons to format the words and
the changes are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to add Markdown syntax to the word to indicate which words
and phrases should look different from each other
Rich Text Editor supports markdown editing when the editorMode set as
**markdown** and using both *keyboard interaction* and *toolbar action*, you
can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation,
[sample link] (https://ej2.syncfusion.com/home/).
The third-party library <b>Marked</b> is used in this sample to convert
markdown into HTML content
        </div>
    </div>
<style>
    .e-richtexteditor textarea.e-content {
        float: left;
        border-right: 1px solid rgba(0, 0, 0, 0.12);
    }
    .e-richtexteditor .e-rte-content .e-content{
        min-height: 150px;
    }

    .e-richtexteditor .e-rte-content {
        overflow: hidden;
    }

    .e-icon-btn.e-active .e-md-preview::before {
        content: '\e350';
    }

    .e-icon-btn .e-md-preview::before {
        content: '\e345';
    }

```

```

    }

    .e-rte-content .e-content {
        float: right;
        width: 50%;
        overflow: auto;
        height: inherit;
        padding: 8px;
        height: 100%;
    }

    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }

    .highcontrast .e-richtexteditor textarea.e-content {
        border-right: 1px solid #fff;
    }

    .sb-header {
        z-index: 100;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to integrate the third party library](#)
- [How to change the editor mode](#)

File browser in EJ2 JavaScript Rich text editor control

Rich Text Editor allows to browse and insert an image in the edit panel using the file browser. File browser allows the users to browse and select a file or folder from the file system and it supports various cloud services.

Required additional package styles and scripts reference

Following packages style and script (local or CDN) reference additionally required to use the file browser feature in Rich Text Editor.

- ej2-data
- ej2-layouts
- ej2-grids
- ej2-filemanager

Map the above packages style and script reference in sample as like below

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Rich Text Editor</title>
<!-- Essential JS 2 Rich Text Editor's dependent material themes -->
<link href="https://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-layouts/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="https://cdn.syncfusion.com/ej2/ej2-filemanager/styles/material.css" rel="stylesheet"
type="text/css" />
<!-- Essential JS 2 Rich Text Editor material theme -->
<link href="https://cdn.syncfusion.com/ej2/ej2-richtexteditor/styles/material.css" rel="stylesheet"
type="text/css" />
<!-- Essential JS 2 Rich Text Editor's dependent scripts -->
<script src="https://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/ej2-splitbuttons/dist/global/ej2-splitbuttons.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/ej2-layouts/dist/global/ej2-layouts.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/ej2-filemanager/dist/global/ej2-filemanager.min.js"
type="text/javascript"></script>

<!-- Essential JS 2 Rich Text Editor's global script -->

<script src="https://cdn.syncfusion.com/ej2/ej2-richtexteditor/dist/global/ej2-richtexteditor.min.js"
type="text/javascript"></script>

</head>

<body>

</body>

</html>

```

The following example explains about how to configure the file browser within the Rich Text Editor component.

- Configure the **FileManager** toolbar item in the **toolbarSettings** API items property.
- Set **enable** property as true on **fileManagerSettings** property to make the file browser in the Rich Text Editor to appear on the **FileManager** toolbar click action.

INDEX.JS

```

/**
 * Rich Text Editor - File browser sample
 */
var hostUrl = 'https://ej2-aspcore-service.azurewebsites.net/';
var defaultRTE = new ej.richtexteditor.RichTextEditor({
  fileManagerSettings: {
    enable: true,
    path: '/Pictures/Food',
    ajaxSettings: {
      url: hostUrl + 'api/FileManager/FileOperations',
      getImageUrl: hostUrl + 'api/FileManager/GetImage',
      uploadUrl: hostUrl + 'api/FileManager/Upload',
      downloadUrl: hostUrl + 'api/FileManager/Download'
    }
  }
});

```



```

    },
    toolbarSettings: {
        items: ['FileManager']
    }
});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
filemanager/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
    </div>
  </div>

```

```

        <ul>
            <li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks,
uploads, etc.</p></li>
        </ul>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Iframe in EJ2 JavaScript Rich text editor control

When the [iframeSettings](#) option is enabled, the Rich Text Editor creates the iframe element as the content area on control initialization; it is used to display and editing the content. In Content area, the editor displays only the body tag of a `<iframe>` document.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
// initialize Rich Text Editor component
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    iframeSettings: {
        enable: true
    }
});
// render initialized Rich Text Editor
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
                <li><p>Capable of handling markdown editing.</p></li>
                <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
                <li><p>Provides a fully customizable toolbar.</p></li>
                <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
                <li><p>Supports third-party library integration.</p></li>
                <li><p>Allows preview of modified content before saving
it.</p></li>
                <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before
    {
        content: "\e726";
    }
    </style>

```

```

    }
    </style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Iframe attributes

The editor allows you to pass an additional attribute to body tag of a `<iframe>` element using [attributes](#) fields of [iframeSettings](#) property. The property contains name/value pairs in string format. It is used to override the default appearance of the content area.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
// initialize Rich Text Editor component
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    iframeSettings: {
        enable: true,
        attributes: {
            readonly: 'readonly'
        }
    }
});
// render initialized Rich Text Editor
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62;; and &#60;DIV&#62;;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before
    {
        content: "\e726";
    }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding external CSS/Script File

The editor offers you to add external CSS file to style the `<iframe>` element. Easily change the appearance of editor's content using an external CSS file using `styles` field in `iframeSettings` property.

Likewise, add the external script file to the `<iframe>` element using `scripts` field of `iframeSettings` to provide the additional functionalities to the Rich Text Editor.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
// initialize Rich Text Editor component
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    iframeSettings: {
        enable: true,
        attributes: {
            readonly: 'readonly'
        },
        resources: {
            scripts: ['my-script.js'],
            styles: ['my-style.css']
        }
    }
});
// render initialized Rich Text Editor
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before
    {
        content: "\e726";
    }
    </style>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to change the editor mode](#)

Format Painter in EJ2 JavaScript Rich Text Editor Control | Syncfusion

A format painter is a tool that allows you to copy the formatting from a piece of text and apply it to another one. Format Painter can be accessed via the toolbar or the keyboard shortcuts. The format painter can copy the formatting of a single word or a whole paragraph. The format painter can be customized using the [formatPainterSettings](#) property.

Enabling the toolbar option for Format Painter

You can add the **FormatPainter** tool in the Rich Text Editor using the **toolbarSettings** [items](#) property.

By double-clicking the format painter toolbar button, **sticky mode** will be enabled. In sticky mode, the format painter will be disabled when the user clicks the **Escape** key again.

The following code example shows how to add the format painter tool in the Rich Text Editor.

INDEX.JSS

```
/**
 * Rich Text Editor - Format Painter sample
 */
var defaultRTE = new ej.richtexteditor.RichTextEditor({
  toolbarSettings: {
    items: ['FormatPainter', 'ClearFormat', 'Bold', 'Italic',
'Underline', '|', 'Formats', 'Alignments',
'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', '|',
'SourceCode', 'Undo', 'Redo']
  },
});
defaultRTE.appendTo('#formatPainterRTE');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Javascript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div id='formatPainterRTE'>
      <h3><strong>Format Painter</strong></h3>
      <p>
        A Format Painter is a Rich Text Editor feature allowing users to quickly
        <span style="background-color: rgb(198, 140, 83);"><strong>copy</strong></span>
        and
        <span style="background-color: rgb(198, 140, 83);"><strong>paste</strong></span>
        formatting from one text to another. With a rich text editor, utilize the format painter as follows:
      </p>
      <ul>
        <li>
          Select the text whose format you want to copy.
        </li>
        <li>
          Click on the <strong><em>Format Painter</em></strong> button in the toolbar. It may look like a paintbrush icon.
        </li>
        <li>
          The cursor will change to a <strong>paintbrush</strong> icon. Click and drag the cursor over the text you want to apply the copied format.
        </li>
        <li>
          Release the mouse button to apply the format.
        </li>
      </ul>
      <p>
        Using the format painter in a rich text editor can save you time when formatting a large document, You can quickly copy and apply formatting to <span style="background-color: rgb(198, 140, 83);"><strong>multiple sections</strong></span>.
        It's a helpful tool for anyone who works with text editing regularly, such as writers, editors, and content creators.
      </p>
    </div>
  </div>
</body>
</html>
<script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
</script>

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Customization of copy and paste format

You can customize the format painter tool in the Rich Text Editor using the `formatPainterSettings` property.

The `allowedFormats` property helps you to specify tag names that allow the formats to be copied from the selected text. For instance, you can include formats from the selected text using tags like `p`; `h1`; `h2`; `h3`; `div`; `ul`; `ol`; `li`; `span`; `strong`; `em`; `code`. The following example demonstrates how to customize this functionality.

Similarly, with the `deniedFormats` property, you can utilize the selectors to prevent specific formats from being pasted onto the selected text. The table below illustrates the selectors and their respective usage.

Type	Description	Selector	Usage
()	Class Selector	<code>h3(e-rte-block-blue-text)</code>	The class name <code>e-rte-block-blue-text</code> of H3 element is not copied.
[]	Attribute Selector	<code>span\[title]</code>	The title attribute of span element is not copied.
{ }	Style Selector	<code>span{background-color, color}</code>	The background-color and color styles of span element is not copied.

Using the `deniedFormats` property following styles are denied copying from the selected text such as `h3(e-rte-block-blue-text){background-color,padding}[title]`; `li{color}`; `span(e-inline-text-highlight){color}[title]`; `strong{color}(e-rte-strong-bg)`.

INDEX.JS

```

/**
 * Rich Text Editor - Format Painter sample
 */
var defaultRTE = new ej.richtexteditor.RichTextEditor({
  toolbarSettings: {
    items: ['FormatPainter', 'ClearFormat', 'Bold', 'Italic',
    'Underline', '|', 'Formats', 'Alignments',
    'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', '|',
    'SourceCode', 'Undo', 'Redo']
  },
  formatPainterSettings: {
    allowedFormats: 'p;h1;h2;h3;div;ul;ol;li;span;strong;em;code;',
    deniedFormats: 'h3(e-rte-block-blue-text){background-
    color,padding,color}[title]; li{color}; span(e-inline-text-
    highlight){color}[title]; strong{color}(e-rte-strong-bg);',
  }
});

```

```
});
defaultRTE.appendTo('#formatPainterRTE');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Javascript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
  </head>
  <body>
    <div id="container">
      <div id='formatPainterRTE'>
        <h3 class="e-rte-block-blue-text" title="Format Painter"
style="color: #0079f3; background-color: #eff6ff; padding: 10px;"><strong>
>Format Painter</strong></h3>
        <p>
          A Format Painter is a Rich Text Editor feature allowing
users to quickly
          <span class="e-inline-text-highlight" style="color:
blue;" title="Styled by CSS Class selector"><strong>copy</strong></span>
          and
          <span class="e-inline-text-highlight" style="color:
blue;" title="Styled by CSS Class selector"><strong>paste</strong></span>
          formatting from one text to another. With a rich text
editor, utilize the format painter as follows:
        </p>
        <ul>
          <li style="color: crimson;">
            Select the text whose format you want to copy.

```

```

        </li>
        <li style="color: crimson;">
            Click on the <strong><em>Format
Painter</em></strong> button in the toolbar. It may look like a paintbrush
icon.
        </li>
        <li style="color: crimson;">
            The cursor will change to a
<strong>paintbrush</strong> icon. Click and drag the cursor over the text
you want to apply the copied format.
        </li>
        <li style="color: crimson;">
            Release the mouse button to apply the format.
        </li>
    </ul>
    <p>
        Using the format painter in a rich text editor can save
you time when formatting a large document, You can quickly
copy and apply formatting
to <strong class="e-rte-strong-bg" style="color:
blue">multiple sections</strong>.
        It's a helpful tool for anyone who works with text
editing regularly, such as writers, editors, and content creators.
    </p>
</div>
</div>
<script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Using the shortcut key to copy and paste the format

You can use the following shortcut keys to copy and paste the format in the Rich Text Editor.

Actions	Keyboard shortcuts	Description
Copy the format	Alt + Shift + c	Copy the selection format or current range.
Pate the format	Alt + Shift + v	Paint the copied format.
Escape	Esc	Remove the previously copied format and disable the sticky mode.

The format painter retains the formatting after application making it possible to apply the same formatting multiple times by using the Alt + Shift + v keyboard shortcut.

Additionally, You can perform the format painter actions programmatically using the [executeCommand](#) public method.

Form support in EJ2 JavaScript Rich text editor control

The following sample demonstrates how to get the Rich Text Editor value in button click.

Render the Rich Text Editor

Render the Rich Text Editor in form.

```
`ts
<form id="myForm" class="form-vertical">
<div class="form-group">
<textarea id="defaultRTE" name="defaultRTE" required maxlength="100" minlength="20" data-msg-
containerid="dateError"> </textarea>
<div id="dateError" style="padding-top: 10px"></div>
</div>
<div style="text-align: center">
<button id="validateSubmit" class="samplebtn e-control e-btn" type="submit" data-
ripple="true">Submit</button>
<button id="resetbtn" class="samplebtn e-control e-btn" type="reset" data-
ripple="true">Reset</button>
</div>
</form>
`
```

Obtain the value

Upon submitting the form, the `getValue` method will be triggered. Through the `FormData` class, get the Rich Text Editor value.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
import { FormValidator } from '@syncfusion/ej2-inputs';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({ showCharCount: true,
maxLength: 100, placeholder: 'Type something' });
defaultRTE.appendTo('#defaultRTE');
let formObject = new FormValidator('#form-element');
document.getElementById('validateSubmit').onclick = function () {
    getValue();
}
function getValue() {
    let form = document.getElementById('form-element');
    let formData = new FormData(form);
    let rteValue = formData.get('defaultRTE');
    alert(rteValue); //Store the value to the data base.
}
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="content" class="box-form" style="margin: 0 auto;
width:750px; padding:25px">

      <form id="form-element" class="form-vertical">
        <div class="form-group">
          <textarea id="defaultRTE" required="" name="defaultRTE"
class="form-control"></textarea>
          <div id="dateError" style="padding-top: 10px"></div>
        </div>
        <div class="form-group">
          <div class="col-sm-3 control-label"></div>
          <div class="col-sm-6">
            <div id="error"></div>
          </div>
        </div>
        <div style="text-align: center">
          <button id="validateSubmit" class="samplebtn e-control
e-btn">Submit</button>
          <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>

```

```

        </div>
    </form>

    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to integrate the third party library](#)
- [How to validate the value](#)

Validation in EJ2 JavaScript Rich text editor control

Validation rules

The Rich Text Editor provides the functionality of character count and its validation. So, you can validate the Rich Text Editor's value on form submission by applying validationRules and validationMessage to the Rich Text Editor.

Rules	Description
required	Requires value for the Rich Text Editor control.
minlength	Requires the value to be of given minimum characters count.
maxlength	Requires the value to be of given maximum characters count.

This sample is used to validate form using the obtrusive Validation. Type the values in Rich Text Editor and the form enables the validation with the formvalidator rules by clicking on the submit externally. All rules are validated by the formvalidator rules.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
import { FormValidator } from '@syncfusion/ej2-inputs';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar);
import { Button } from '@syncfusion/ej2-buttons';
let button: Button;
let defaultRTE: RichTextEditor = new RichTextEditor({ showCharCount: true,
maxLength: 100, placeholder: 'Type something',

```

```

change : function() {
    button.disabled = false;
}
});
defaultRTE.appendTo('#defaultRTE');
let dialog: Dialog;
button = new Button({
disabled :true
});
button.appendTo('#validateSubmit');
new FormValidator('#form-element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="content" class="box-form" style="margin: 0 auto;
width:750px; padding:25px">
            <form id="form-element" class="form-vertical">
                <div class="form-group">
                    <textarea id="defaultRTE" name="defaultRTE" required=""
maxlength="100" minlength="20" data-msg-containerid="dateError">
</textarea>

                    <div id="dateError" style="padding-top: 10px"></div>
                </div>

```



```

        <div style="text-align: center">
            <button id="validateSubmit">Submit</button>
            <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
        </div>
    </form>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Validation message

The default error message for a rule can be customizable by defining it along with the concern rule object as follows.

In the following sample, customize the error message along with the concern rule.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar);
import { Button } from '@syncfusion/ej2-buttons';
let button: Button;
let defaultRTE: RichTextEditor = new RichTextEditor({ showCharCount: true,
maxLength: 100, placeholder: 'Type something' ,
change : function() {
    button.disabled = false;
} });
defaultRTE.appendTo('#defaultRTE');
button = new Button({
disabled :true
});
button.appendTo('#validateSubmit');
let option: FormValidatorModel = {
    rules: {
        // Initialize the CustomPlacement.
        defaultRTE: {
            required: true,
            minLength: [20, 'Need atleast 6 character length'],
            maxLength:[100, 'Maximum 100 character only']
        }
    },
    customPlacement: (inputElement: HTMLElement, dateError: HTMLElement)=>{
        document.getElementById('dateError').appendChild(dateError);
    }
}

```

```
};
let formObject: FormValidator = new FormValidator('#form-element', option);
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="content" class="box-form" style="margin: 0 auto;
width:750px; padding:25px">

      <form id="form-element" class="form-vertical">
        <div class="form-group">
          <textarea id="defaultRTE" required="" name="defaultRTE"
class="form-control"></textarea>
          <div id="dateError" style="padding-top: 10px"></div>
        </div>
        <div class="form-group">
          <div class="col-sm-3 control-label"></div>
          <div class="col-sm-6">
            <div id="error"></div>
          </div>
        </div>
        <div style="text-align: center">
          <button id="validateSubmit" class="samplebtn e-control
e-btn">Submit</button>
```

```

        <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
        </div>
    </form>

    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom placement of validation message

The FormValidator has an event [customPlacement](#) which can be used to place the error message from default position to desired custom location.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, Count,
QuickToolbar);
import { Button } from '@syncfusion/ej2-buttons';
let button: Button;
let defaultRTE: RichTextEditor = new RichTextEditor({ showCharCount: true,
maxLength: 100, placeholder: 'Type something' ,
change : function() {
    button.disabled = false;
} });
defaultRTE.appendTo('#defaultRTE');
button = new Button({
disabled :true
});
button.appendTo('#validateSubmit');
let option: FormValidatorModel = {
    rules: {
        // Initialize the CustomPlacement.
        defaultRTE: { required: [true, 'RTE: value is required'], minLength: [15,
'RTE: Need atleast 6 character length'], maxLength:[100, 'RTE: Maximum 100
character only'] }
    },
    customPlacement: (inputElement: HTMLElement, error: HTMLElement)=>{
        document.getElementById('error').appendChild(error);
    }
};
let formObject: FormValidator = new FormValidator('#form-element', option);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="content" class="box-form" style="margin: 0 auto;
width:750px; padding:25px">

      <form id="form-element" class="form-vertical">
        <div class="form-group">
          <textarea id="defaultRTE" required="" name="defaultRTE"
class="form-control"></textarea>
          <div id="dateError" style="padding-top: 10px"></div>
        </div>
        <div class="form-group">
          <div class="col-sm-3 control-label"></div>
          <div class="col-sm-6">
            <div id="error"></div>
          </div>
        </div>
        <div style="text-align: center">
          <button id="validateSubmit" class="samplebtn e-control
e-btn">Submit</button>
          <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
        </div>
      </form>

    </div>
  </div>

```

```

    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style in EJ2 JavaScript Rich text editor control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the Rich Text Editor's content

Use the following CSS to customize the default Rich Text Editor's content properties like font-family, font-size and color.

`css

/ To change font family and font size /

```

.e-richtexteditor .e-rte-content .e-content,
.e-richtexteditor .e-source-content .e-content {
font-size: 20px;
font-family: Segoe ui;
}

```

/ To change font color and content background /

```

.e-richtexteditor .e-rte-content,
.e-richtexteditor .e-source-content {
background: seashell;
color: blue;
}

```

`

Customizing the Rich Text Editor's toolbar

Use the following CSS to customize the default color in the Rich Text Editor's toolbar icon.

`css

/ To change font color for toolbar icon /

```

.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons,
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons:active {
color: red;
}

```

/ To change font color for toolbar button /

```
.e-toolbar .e-tbar-btn,
.e-toolbar .e-tbar-btn:active,
.e-toolbar .e-tbar-btn:hover {
color: red;
}
```

/ To change font color for toolbar button in active state/

```
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active .e-icons, .e-richtexteditor .e-
rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active .e-rte-dropdown-btn-text {
color: red;
}
```

/ To change font color for expanded toolbar items /

```
.e-richtexteditor .e-rte-toolbar .e-toolbar-extended .e-toolbar-item .e-tbar-btn .e-icons,
.e-toolbar.e-extended-toolbar .e-toolbar-extended .e-toolbar-item .e-tbar-btn {
color: red;
}
```

`

Customizing the Rich Text Editor's character count

Use the following CSS to customize the default color in the Rich Text Editor's character count.

`css

/ To change font color, font family, font size and opacity /

```
.e-richtexteditor .e-rte-character-count {
color: red;
font-family: segoe ui;
font-size: 18px;
opacity: 00.54;
padding-bottom: 2px;
padding-right: 14px;
}
```

`

Globalization in EJ2 JavaScript Rich text editor control

Localization

The Rich Text Editor provides an option to localize its strings; it is used to adapting the editor to a particular local language. By default, the editor will use the **US English (en-US)** as its language. Please find the table with a list of keys and their corresponding values for the default language (en-US).

```
`ts
'en-US': {
  'richtexteditor': {
    'alignments': 'Alignments',
    'justifyleft': 'JustifyLeft',
    'justifycenter': 'JustifyCenter',
    'justifyright': 'Justify Right',
    'justifyfull': 'Justify Full',
    'formats': 'Formats',
    'fontname': 'Font Name',
    'fontsize': 'Font Size',
    'fontcolor': 'Font Color',
    'backgroundcolor': 'Background color',
    'bold': 'Bold',
    'italic': 'Italic',
    'underline': 'Underline',
    'strikethrough': 'Strikethrough',
    'clearall': 'Clear All',
    'clearformat': 'Clear Format',
    'cut': 'Cut',
    'copy': 'Copy',
    'paste': 'Paste',
    'unorderedlist': 'unordered list',
    'orderedlist': 'ordered list',
    'indent': 'Increase Indent',
    'outdent': 'Decrease Indent',
    'undo': 'Undo',
    'redo': 'Redo',
    'superscript': 'Superscript',
    'subscript': 'Subscript',
    'createlink': 'Insert link',
    'removelink': 'remove link',
    'openlink': 'Open link',
```

'editlink': 'Edit link',
'image': 'Insert image',
'replace': 'Replace',
'align': 'Align',
'caption': 'Caption',
'remove': 'Remove',
'insertlink': 'insertlink',
'display': 'Display',
'alttext': 'Alternative Text',
'dimension': 'Dimension',
'fullscreen': 'Full Screen',
'maximize': 'Maximize',
'minimize': 'Minimize',
'zoomin': 'Zoom In',
'zoomout': 'Zoom Out',
'uppercase': 'Upper Case',
'lowercase': 'Lower Case',
'print': 'Print',
'sourcecode': 'Source Code',
'preview': 'Preview',
'viewside': 'View Side',
'insertcode': 'Insert Code',
'linkText': 'Display Text',
'linkTooltipLabel': 'Title',
'linkWebUrl': 'Web Address',
'linkTitle': 'Enter a title',
'linkOpenInNewWindow': 'Open Link in New Window',
'linkHeader': 'Insert Link',
'dialogInsert': 'Insert',
'dialogCancel': 'Cancel',
'dialogUpdate': 'Update',
'imageHeader': 'Insert Image',
'imageLinkHeader': 'You can also provide a link from the web',

'mdimageLink': 'Please provide a URL for your image',
'imageUploadMessage': 'Drop image here or browse to upload',
'imageDeviceUploadMessage': 'Click here to upload',
'imageAlternateText': 'Alternate Text',
'alternateHeader': 'Alternative Text',
'browse': 'Browse',
'imageCaption': 'Caption',
'imageSizeHeader': 'Image Size',
'imageHeight': 'Height',
'imageWidth': 'Width',
'textPlaceholder': 'Enter Text',
'inserttablebtn': 'Insert Table',
'tabledialogHeader': 'Insert Table',
'tableWidth': 'Width',
'cellpadding': 'Cell Padding',
'cellspacing': 'Cell Spacing',
'columns': 'Number of columns',
'rows': 'Number of rows',
'tableRows': 'Table Rows',
'tableColumns': 'Table Columns',
'tableCellHorizontalAlign': 'Table Cell Horizontal Align',
'tableCellVerticalAlign': 'Table Cell Vertical Align',
'createTable': 'Create Table',
'removeTable': 'Remove Table',
'tableHeader': 'Table Header',
'tableRemove': 'Table Remove',
'tableCellBackground': 'Table Cell Background',
'tableEditProperties': 'Table Edit Properties',
'styles': 'Styles',
'insertColumnLeft': 'Insert Column Left',
'insertColumnRight': 'Insert Column Right',
'deleteColumn': 'Delete Column',
'insertRowBefore': 'Insert Row Before',

'insertRowAfter': 'Insert Row After',
'deleteRow': 'Delete Row',
'tableEditHeader': 'Edit Table',
'TableHeadingText': 'Heading',
'TableColText': 'Col',
'imageInsertLinkHeader': 'Insert Link',
'editImageHeader': 'Edit Image',
'alignmentsDropDownLeft': 'Align Left',
'alignmentsDropDownCenter': 'Align Center',
'alignmentsDropDownRight': 'Align Right',
'alignmentsDropDownJustify': 'Align Justify',
'imageDisplayDropDownInline': 'Inline',
'imageDisplayDropDownBreak': 'Break',
'tableInsertRowDropDownBefore': 'Insert row before',
'tableInsertRowDropDownAfter': 'Insert row after',
'tableInsertRowDropDownDelete': 'Delete row',
'tableInsertColumnDropDownLeft': 'Insert column left',
'tableInsertColumnDropDownRight': 'Insert column right',
'tableInsertColumnDropDownDelete': 'Delete column',
'tableVerticalAlignDropDownTop': 'Align Top',
'tableVerticalAlignDropDownMiddle': 'Align Middle',
'tableVerticalAlignDropDownBottom': 'Align Bottom',
'tableStylesDropDownDashedBorder': 'Dashed Borders',
'tableStylesDropDownAlternateRows': 'Alternate Rows',
'pasteFormat': 'Paste Format',
'pasteFormatContent': 'Choose the formatting action',
'plainText': 'Plain Text',
'cleanFormat': 'Clean',
'keepFormat': 'Keep',
'formatsDropDownParagraph': 'Paragraph',
'formatsDropDownCode': 'Code',
'formatsDropDownQuotation': 'Quotation',
'formatsDropDownHeading1': 'Heading 1',

```

'formatsDropDownHeading2': 'Heading 2',
'formatsDropDownHeading3': 'Heading 3',
'formatsDropDownHeading4': 'Heading 4',
'fontNameSegoeUI': 'Segoe UI',
'fontNameArial': 'Arial',
'fontNameGeorgia': 'Georgia',
'fontNameImpact': 'Impact',
'fontNameTahoma': 'Tahoma',
'fontNameTimesNewRoman': 'Times New Roman',
'fontNameVerdana': 'Verdana',
'numberFormatListNumber': 'Number',
'numberFormatListLowerAlpha': 'LowerAlpha',
'numberFormatListUpperAlpha': 'UpperAlpha',
'numberFormatListLowerRoman': 'LowerRoman',
'numberFormatListUpperRoman': 'UpperRoman',
'numberFormatListLowerGreek': 'LowerGreek',
'bulletFormatListDisc': 'Disc',
'bulletFormatListCircle': 'Circle',
'bulletFormatListSquare': 'Square',
'numberFormatListNone': 'None',
'bulletFormatListNone': 'None',
'formatPainter': 'Format Painter',
'emojiPicker': 'Emoji Picker',
'embeddedCode': 'Embedded Code',
'pasteEmbeddedCodeHere': 'Paste Embedded Code here',
'emojiPickerTypeToFind': 'Type to find',
'emojiPickerNoResultFound': 'No results found',
'emojiPickerTrySomethingElse': 'Try something else',
}
}
,

```

To localize the editor's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the editor in German language ("de-DE").

```
`ts
'de-DE': {
  'richtexteditor': {
    'alignments': 'Alignments',
    'justifyLeft': 'Ausrichten von Text links',
    'justifyCenter': 'Text-Zentrum',
    'justifyRight': 'Ausrichten von Text rechts',
    'justifyFull': 'rechtfertigen',
    'fontName': 'Wählen Sie Schriftfamilie',
    'fontSize': 'Wählen Sie Schriftgröße',
    'fontColor': 'Wählen Sie die Farbe',
    'backgroundColor': 'Hintergrundfarbe',
    'bold': 'fett',
    'italic': 'kursiv',
    'underline': 'unterstreichen',
    'strikethrough': 'Durchgestrichen',
    'clearAll': 'Alles',
    'clearFormat': 'Klar Format',
    'cut': 'schneiden',
    'copy': 'Kopieren',
    'paste': 'Paste',
    'unorderedList': 'Legen Sie ungeordnete Liste',
    'orderedList': 'Geordnete Liste einfügen',
    'indent': 'Einzug',
    'outdent': 'Einzug verkleinern',
    'undo': 'lösen',
    'redo': 'Wiederherstellen',
    'superscript': 'Überschrift',
    'subscript': 'index',
    'createLink': 'Link einfügen',
    'removeLink': 'fjern Hyperlink',
    'openLink': 'Open link',
    'editLink': 'Edit link',
```

'image': 'Bild einfügen',
'replace': 'ersetzen',
'align': 'ausrichten',
'caption': 'Bildbeschriftung',
'formats': 'Formats',
'remove': 'Löschen',
'insertLink': 'Link einfügen',
'display': 'Anzeige',
'alttext': 'alternativer Text',
'dimension': 'Größe',
'fullscreen': 'Vollbild',
'maximize': 'Maximieren',
'minimize': 'minimieren',
'zoomIn': 'hineinzoomen',
'zoomOut': 'Rauszoomen',
'upperCase': 'Großbuchstaben',
'lowerCase': 'Kleinbuchstaben',
'print': 'Drucken',
'sourcecode': 'Quellcode',
'preview': 'Vorschau',
'viewside': 'Seite anzeigen',
'insertcode': 'Code eingeben',
'linkText': 'Displaytekst',
'linkTooltipLabel': 'tooltip',
'linkWebUrl': 'Webadres',
'linkOpenInNewWindow': 'Open de link in een nieuw venster',
'linkHeader': 'Link invoegen',
'dialogInsert': 'invoegen',
'dialogCancel': 'Annuleer',
'dialogUpdate': 'Bijwerken',
'imageHeader': 'Voeg afbeelding in',
'imageLinkHeader': 'U kunt ook een link van internet opgeven',
'imageUploadMessage': 'Zet hier een afbeelding neer of klik om te uploaden',

'imageDeviceUploadMessage': 'Klik hier om te uploaden',
'imageAlternateText': 'Alternatieve tekst',
'alternateHeader': 'Alternatieve tekst',
'browse': 'Blader',
'imageUrl': 'URL',
'imageCaption': 'onderschrift',
'imageSizeHeader': 'Afbeeldingsgrootte',
'imageHeight': 'Hoogte',
'imageWidth': 'Breedte',
'textPlaceholder': 'Text eingeben',
'inserttablebtn': 'Tabelle einfügen',
'tableDialogHeader': 'Tabelle einfügen',
'tableWidth': 'Breite',
'cellpadding': 'Zellauffüllung',
'cellspacing': 'Zellabstand',
'columns': 'Anzahl der Spalten',
'rows': 'Reihenanzahl',
'tableRows': 'Tabellenzeilen',
'tableColumns': 'Tabellenspalten',
'tableCellHorizontalAlign': 'Horizontale Ausrichtung der Tabellenzelle',
'tableCellVerticalAlign': 'Vertikale Ausrichtung der Tabellenzelle',
'createTable': 'Tabelle erstellen',
'removeTable': 'Tabelle entfernen',
'tableHeader': 'Tabellenkopfzeile',
'tableRemove': 'Tabelle entfernen',
'tableCellBackground': 'Tabellenzellenhintergrund',
'tableEditProperties': 'Eigenschaften der Tabellenbearbeitung',
'styles': 'Styles',
'insertColumnLeft': 'Spalte links einfügen',
'insertColumnRight': 'Spalte rechts einfügen',
'deleteColumn': 'Spalte löschen',
'insertRowBefore': 'Zeile vor einfügen',
'insertRowAfter': 'Zeile einfügen nach',

'deleteRow': 'Zeile löschen',
'tableEditHeader': 'Tabelle bearbeiten',
'TableHeadingText': 'Überschrift',
'TableColText': 'Col',
'imageInsertLinkHeader': 'Link einfügen',
'editImageHeader': 'Bild bearbeiten',
'alignmentsDropDownLeft': 'Linksbündig',
'alignmentsDropDownCenter': 'Im Zentrum anordnen',
'alignmentsDropDownRight': 'Rechts ausrichten',
'alignmentsDropDownJustify': 'Justize ausrichten',
'imageDisplayDropDownInline': 'In der Reihe',
'imageDisplayDropDownBreak': 'Brechen',
'tableInsertRowDropDownBefore': 'Reihe vorher einfügen',
'tableInsertRowDropDownAfter': 'Zeile danach einfügen',
'tableInsertRowDropDownDelete': 'Zeile löschen',
'tableInsertColumnDropDownLeft': 'Spalte links einfügen',
'tableInsertColumnDropDownRight': 'Spalte rechts einfügen',
'tableInsertColumnDropDownDelete': 'Spalte löschen',
'tableVerticalAlignDropDownTop': 'Top ausrichten',
'tableVerticalAlignDropDownMiddle': 'Mitte ausrichten',
'tableVerticalAlignDropDownBottom': 'Unten ausrichten',
'tableStylesDropDownDashedBorder': 'Gestrichelte Grenzen',
'tableStylesDropDownAlternateRows': 'Alternative Zeilen',
'pasteFormat': 'Format einfügen',
'pasteFormatContent': 'Wählen Sie die Formatierungsaktion aus',
'plainText': 'Einfacher Text',
'cleanFormat': 'sauber',
'keepFormat': 'Behalten',
'formatsDropDownParagraph': 'Absatz',
'formatsDropDownCode': 'Kodex',
'formatsDropDownQuotation': 'Zitat',
'formatsDropDownHeading1': 'Überschrift 1',
'formatsDropDownHeading2': 'Überschrift 2',

```

'formatsDropDownHeading3': 'Überschrift 3',
'formatsDropDownHeading4': 'Überschrift 4',
'fontNameSegoeUI': 'Segoe UI',
'fontNameArial': 'Arial',
'fontNameGeorgia': 'Georgia',
'fontNameImpact': 'Einschlag',
'fontNameTahoma': 'Tahoma',
'fontNameTimesNewRoman': 'Mal Neu römisch',
'fontNameVerdana': 'Verdana'
}
},
`

```

The below sample demonstrate that, the Rich Text Editor control rendered with 'de-DE' German language using [locale](#) property.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { L10n } from '@syncfusion/ej2-base';
enableRipple(true);
L10n.load({
  'de-DE': {
    'richtexteditor': {
      'alignments': 'Alignments',
      'justifyLeft': 'Ausrichten von Text links',
      'justifyCenter': 'Text-Zentrum',
      'justifyRight': 'Ausrichten von Text rechts',
      'justifyFull': 'rechtfertigen',
      'fontName': 'Wählen Sie Schriftfamilie',
      'fontSize': 'Wählen Sie Schriftgröße',
      'fontColor': 'Wählen Sie die Farbe',
      'backgroundColor': 'Hintergrundfarbe',
      'bold': 'fett',
      'italic': 'kursiv',
      'underline': 'unterstreichen',
      'strikethrough': 'Durchgestrichen',
      'clearAll': 'Alles',
      'clearFormat': 'Klar Format',
      'cut': 'schneiden',
      'copy': 'Kopieren',
      'paste': 'Paste',
      'unorderedList': 'Legen Sie ungeordnete Liste',
      'orderedList': 'Geordnete Liste einfügen',
      'indent': 'Einzug',
      'outdent': 'Einzug verkleinern',
      'undo': 'lösen',
      'redo': 'Wiederherstellen',
      'superscript': 'Überschrift',

```



```

    'subscript': 'index',
    'createLink': 'Link einfügen',
    'removeLink': 'fjern Hyperlink',
    'openLink': 'Open link',
    'editLink': 'Edit link',
    'image': 'Bild einfügen',
    'replace': 'ersetzen',
    'align': 'ausrichten',
    'caption': 'Bildbeschriftung',
    'formats': 'Formats',
    'remove': 'Löschen',
    'insertLink': 'Link einfügen',
    'display': 'Anzeige',
    'alttext': 'alternativer Text',
    'dimension': 'Größe',
    'fullscreen': 'Vollbild',
    'maximize': 'Maximieren',
    'minimize': 'minimieren',
    'zoomIn': 'hineinzoomen',
    'zoomOut': 'Rauszoomen',
    'upperCase': 'Großbuchstaben',
    'lowerCase': 'Kleinbuchstaben',
    'print': 'Drucken',
    'sourcecode': 'Quellcode',
    'preview': 'Vorschau',
    'viewside': 'Seite anzeigen',
    'insertcode': 'Code eingeben',
    'linkText': 'Displaytekst',
    'linkTooltipLabel': 'tooltip',
    'linkWebUrl': 'Webadres',
    'linkOpenInNewWindow': 'Open de link in een nieuw venster',
    'linkHeader': 'Link invoegen',
    'dialogInsert': 'invoegen',
    'dialogCancel': 'Annuleer',
    'dialogUpdate': 'Bijwerken',
    'imageHeader': 'Voeg afbeelding in',
    'imageLinkHeader': 'U kunt ook een link van internet opgeven',
    'imageUploadMessage': 'Zet hier een afbeelding neer of klik om
te uploaden',
    'imageDeviceUploadMessage': 'Klik hier om te uploaden',
    'imageAlternateText': 'Alternatieve tekst',
    'alternateHeader': 'Alternatieve tekst',
    'browse': 'Blader',
    'imageUrl': 'URL',
    'imageCaption': 'onderschrift',
    'imageSizeHeader': 'Afbeeldingsgrootte',
    'imageHeight': 'Hoogte',
    'imageWidth': 'Breedte',
    'textPlaceholder': 'Text eingeben',
    'inserttablebtn': 'Tabelle einfügen',
    'tabledialogHeader': 'Tabelle einfügen',
    'tableWidth': 'Breite',
    'cellpadding': 'Zellauffüllung',
    'cellspacing': 'Zellabstand',
    'columns': 'Anzahl der Spalten',
    'rows': 'Reihenanzahl',
    'tableRows': 'Tabellenzeilen',

```

```

        'tableColumns': 'Tabellenspalten',
        'tableCellHorizontalAlign': 'Horizontale Ausrichtung der
Tabellenzelle',
        'tableCellVerticalAlign': 'Vertikale Ausrichtung der
Tabellenzelle',
        'createTable': 'Tabelle erstellen',
        'removeTable': 'Tabelle entfernen',
        'tableHeader': 'Tabellenkopfzeile',
        'tableRemove': 'Tabelle entfernen',
        'tableCellBackground': 'Tabellenzellenhintergrund',
        'tableEditProperties': 'Eigenschaften der Tabellenbearbeitung',
        'styles': 'Styles',
        'insertColumnLeft': 'Spalte links einfügen',
        'insertColumnRight': 'Spalte rechts einfügen',
        'deleteColumn': 'Spalte löschen',
        'insertRowBefore': 'Zeile vor einfügen',
        'insertRowAfter': 'Zeile einfügen nach',
        'deleteRow': 'Zeile löschen',
        'tableEditHeader': 'Tabelle bearbeiten',
        'TableHeadingText': 'Überschrift',
        'TableColText': 'Col',
        'imageInsertLinkHeader': 'Link einfügen',
        'editImageHeader': 'Bild bearbeiten',
        'alignmentsDropDownLeft': 'Linksbündig',
        'alignmentsDropDownCenter': 'Im Zentrum anordnen',
        'alignmentsDropDownRight': 'Rechts ausrichten',
        'alignmentsDropDownJustify': 'Justize ausrichten',
        'imageDisplayDropDownInline': 'In der Reihe',
        'imageDisplayDropDownBreak': 'Brechen',
        'tableInsertRowDropDownBefore': 'Reihe vorher einfügen',
        'tableInsertRowDropDownAfter': 'Zeile danach einfügen',
        'tableInsertRowDropDownDelete': 'Zeile löschen',
        'tableInsertColumnDropDownLeft': 'Spalte links einfügen',
        'tableInsertColumnDropDownRight': 'Spalte rechts einfügen',
        'tableInsertColumnDropDownDelete': 'Spalte löschen',
        'tableVerticalAlignDropDownTop': 'Top ausrichten',
        'tableVerticalAlignDropDownMiddle': 'Mitte ausrichten',
        'tableVerticalAlignDropDownBottom': 'Unten ausrichten',
        'tableStylesDropDownDashedBorder': 'Gestrichelte Grenzen',
        'tableStylesDropDownAlternateRows': 'Alternative Zeilen',
        'pasteFormat': 'Format einfügen',
        'pasteFormatContent': 'Wählen Sie die Formatierungsaktion aus',
        'plainText': 'Einfacher Text',
        'cleanFormat': 'sauber',
        'keepFormat': 'Behalten',
        'formatsDropDownParagraph': 'Absatz',
        'formatsDropDownCode': 'Kodex',
        'formatsDropDownQuotation': 'Zitat',
        'formatsDropDownHeading1': 'Überschrift 1',
        'formatsDropDownHeading2': 'Überschrift 2',
        'formatsDropDownHeading3': 'Überschrift 3',
        'formatsDropDownHeading4': 'Überschrift 4',
        'fontNameSegoeUI': 'Segoe UI',
        'fontNameArial': 'Arial',
        'fontNameGeorgia': 'Georgia',
        'fontNameImpact': 'Einschlag',
        'fontNameTahoma': 'Tahoma',

```

```

        'fontNameTimesNewRoman': 'Mal Neu römisch',
        'fontNameVerdana': 'Verdana'
    }
});
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({locale: 'de-DE'});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>

```

```

        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

RTL

Specifies the direction of the Rich Text Editor component using the [enableRtl](#) property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

It will not change based on the locale property.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({enableRtl : true });
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }
    </style>

```

```

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[enableRtl](#) property will not change based on [locale](#) property.

Exec command in EJ2 JavaScript Rich text editor control

In Rich Text Editor, `execCommand` used to perform command for the modification of content in editable area.

The `execCommand` will perform the following commands.

Commands	Description	Code snippets
----- ----- -----		
bold	Bold the selected content in the Rich Text Editor.	<code>rteObj.executeCommand('bold');</code>
italic	The selected text will be italics.	<code>rteObj.executeCommand('italic');</code>
underline	Underline the selected text in the Rich Text Editor.	<code>rteObj.executeCommand('underline');</code>
strikeThrough	Apply single line strike through formatting for the selected text.	<code>rteObj.executeCommand('strikeThrough');</code>
superscript	Makes the selected text as superscript (higher).	<code>rteObj.executeCommand('superscript');</code>
subscript	Makes the selected text as subscript (lower).	<code>rteObj.executeCommand('subscript');</code>
uppercase	Change the case of selected text to upper in the content.	<code>rteObj.executeCommand('uppercase');</code>
lowercase	Change the case of selected text to lower in the content.	<code>rteObj.executeCommand('lowercase');</code>
fontColor	Apply the specified font color for the selected text.	<code>rteObj.executeCommand('fontColor', 'yellow');</code>
fontName	Apply the specified font name for the selected text.	<code>rteObj.executeCommand('fontName', 'Arial');</code>
fontSize	Apply the specified font size for the selected text.	<code>rteObj.executeCommand('fontSize', '10pt');</code>
formatBlock	Apply the specified format styles for the selected text.	<code>rteObj.executeCommand('formatBlock', 'H1');</code>

| backColor | Apply the specified background color the selected text. |
`rteObj.executeCommand('backColor', 'red');`|

| justifyCenter | Align the content with center margin. | `rteObj.executeCommand('justifyCenter');`|

| justifyFull | Align the content with justify margin. | `rteObj.executeCommand('justifyFull');`|

| justifyLeft | Align the content with left margin. | `rteObj.executeCommand('justifyLeft');`|

| justifyRight | Align the content with right margin. | `rteObj.executeCommand('justifyLeft');`|

| undo | Allows to undo the actions. | `rteObj.executeCommand('undo');`|

| createLink | Creates a hyperlink to a text or image to a specific location in the content. |
`rteObj.executeCommand('createLink',{ text: 'Links', url: 'http://', title : 'Link' });`|

| indent | Allows to increase the indent level of the content. | `rteObj.executeCommand('indent');`|

| insertHTML | Insert the html content to the current cursor position. |
`rteObj.executeCommand('insertHTML', 'inserted an html');`|

| insertOrderedList | Create a new list item(numbered). |
`rteObj.executeCommand('insertOrderedList');`|

| insertUnorderedList | Create a new list item(bulleted). |
`rteObj.executeCommand('insertUnorderedList');`|

| outdent | Allows to decrease the indent level of the content. |
`rteObj.executeCommand('outdent');`|

| redo | Allows to redo the actions | `rteObj.executeCommand('redo');`|

| removeFormat | remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. |
`rteObj.executeCommand('removeFormat');`|

| insertText | Insert text to the current cursor position. | `rteObj.executeCommand('insertText', 'inserted a text');`|

| insertImage | Insert an image to the current cursor position. |
`rteObj.executeCommand('insertImage', { url: 'https://ej2.syncfusion.com/javascript/demos/src/rich-text-editor/images/RTEImage-Feather.png', cssClass: 'rte-img' });`|

| copyFormatPainter | Copy the format of selected text and apply it to another text. |
`rteObj.executeCommand('copyFormatPainter', formatPainterSettings);`|

| applyFormatPainter | Apply the copied format to the selected text. |
`rteObj.executeCommand('applyFormatPainter');`|

| escapeFormatPainter | Remove the previously copied format and disable the sticky mode |
`rteObj.executeCommand('escapeFormatPainter');`|

Note: The 'ExecuteCommand' public method is not supported in Syncfusion Markdown Editor

Miscellaneous in EJ2 JavaScript Rich text editor control

Placeholder

Specifies the placeholder for the Rich Text Editor's content used when the Rich Text Editor body is empty through the [placeholder](#) property.

Through the `e-rte-placeholder` class to define our custom font family, font color, and styles to the placeholder text.

```
`ts
.e-richtexteditor .e-rte-placeholder {
font-family: monospace;
}
```

The below sample demonstrates the placeholder option in Rich Text Editor.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Count } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Count);
let defaultRTE: RichTextEditor = new RichTextEditor({ placeholder: 'Type
Something' });
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Rich Text Editor</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Character count

The Rich Text Editor automatically counts the number of characters in the content are while typing using the [showCharCount](#) property. The characters count displayed at the bottom of the editor. You can limit the number of characters in your content using the [maxLength](#) property. By default, the editor sets the characters limit value is infinity.

The character count color will be modified based on the characters in the Rich Text Editor.

| Status | Description |

|-----|-----|

| normal | Till 70% of given maxLength count reach, character count color is black. |

| warning | Once the number of character count in the Rich Text Editor reached 70% of given maxLength count, the character count color will be orange, indicating that, the Rich Text Editor value going to reach the maximum count. |

| error | Once the number of character count in the Rich Text Editor reached 90% of given maxLength count, the character count color will be red, indicating that, the Rich Text Editor value reached the maximum count. |

To create Rich Text Editor with [showCharCount](#) feature, inject the Count module to the RTE using the `RichTextEditor.Inject(Count)` method.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Count } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Count);
let defaultRTE: RichTextEditor = new RichTextEditor({ showCharCount: true,
maxLength: 2000 });
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Code view

Rich Text Editor includes the ability for users to directly edit HTML code via **Source View** in the text area. If you made any modification in Source view directly, the changes will be reflected in the Rich Text Editor's content. So, the users will have more flexibility over the content they have created.

INDEX.TS

```
import { addClass, removeClass, Browser } from '@syncfusion/ej2-base';
import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
import { createElement } from '@syncfusion/ej2-base';
let defaultRTE: RichTextEditor = new RichTextEditor({ toolbarSettings: {
  items: ['SourceCode']},
  showCharCount: true,
  actionComplete: actionCompleteHandler});
defaultRTE.appendTo('#defaultRTE');
let myCodeMirror: any;
function mirrorConversion(e?: any): void {
  let id: string = defaultRTE.getID() + 'mirror-view';
  let textArea: HTMLElement = defaultRTE.contentModule.getEditPanel() as
HTMLElement;
  let mirrorView: HTMLElement = defaultRTE.element.querySelector('#' + id)
as HTMLElement;
  let charCount: HTMLElement = defaultRTE.element.querySelector('.e-rte-
character-count') as HTMLElement;
  if (e.targetItem === 'Preview') {
    textArea.style.display = 'block';
    mirrorView.style.display = 'none';
    textArea.innerHTML = myCodeMirror.getValue();
    charCount.style.display = 'block';
  } else {
    if (!mirrorView) {
      mirrorView = createElement('div', { className: 'e-content' });
      mirrorView.id = id;
      textArea.parentNode.appendChild(mirrorView);
    } else {
      mirrorView.innerHTML = '';
    }
    textArea.style.display = 'none';
    mirrorView.style.display = 'block';
    renderCodeMirror(mirrorView, defaultRTE.value);
    charCount.style.display = 'none';
  }
}
function renderCodeMirror(mirrorView: HTMLElement, content: string): void {
  myCodeMirror = CodeMirror(mirrorView, {
    value: content,
    lineNumbers: true,
    mode: 'text/html',
    lineWrapping: true,
  });
}
```

```
function actionCompleteHandler(e: any): void {
    if (e.targetItem && (e.targetItem === 'SourceCode' || e.targetItem === 'Preview')) {
        this.sourceCodeModule.getPanel().style.display = 'none';
        mirrorConversion(e);
    } else {
        setTimeout(() => {
            defaultRTE.toolbarModule.refreshToolbarOverflow(); }, 400);
    }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <!--CodeMirror references-->
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.js"
type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/css/css.js
" type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/xml/xml.js
" type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/htmlmixed/
htmlmixed.js" type="text/javascript"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/javascript
/javascript.js" type="text/javascript"></script>
```

```

<link
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.min
.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id="defaultRTE">
        <p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
        <p><b>Key features:</b></p>
        <ul><li><p>Provides <b>IFRAME</b> and <b>&#62;DIV</b>
modes</p></li>
        <li><p>Capable of handling markdown editing.</p></li>
        <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
        <li><p>Provides a fully customizable toolbar.</p></li>
        <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
        <li><p>Supports third-party library integration.</p></li>
        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
        </ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Undo/Redo Manager

Undo and redo tools allow you to edit the text by disregard/cancel the recently made changes and restore it to previous state. It is a useful tool to restore the performed action which got changed by mistake. By default, upto 30 actions can be undo/redo in the editor.

To undo and redo operations, do one of the following:

- Press the undo/redo button on the toolbar
- Press the **Ctrl + Z**/**Ctrl + Y** combination on the keyboard

Customize the undo/redo step count using [undoRedoSteps](#) property. By default, undo/redo actions take **300ms** time interval for store the action to the **undoRedoManager**. The time interval can be customized by using the [undoRedoTimer](#).

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 340,
    toolbarSettings: {
        items: ['Undo', 'Redo']},
    undoRedoSteps: 50,
    undoRedoTimer: 400
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
```

```

<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
    <p><b>Key features:</b></p>
    <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
    <li><p>Capable of handling markdown editing.</p></li>
    <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
    <li><p>Provides a fully customizable toolbar.</p></li>
    <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
    <li><p>Supports third-party library integration.</p></li>
    <li><p>Allows preview of modified content before saving
it.</p></li>
    <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevention of cross-site scripting (XSS)

The Rich Text Editor allows the users to edit the content with security by preventing cross-site scripting (XSS). By default, provided built-in support to remove the elements from editor content, which cause XSS attack. The editor removes the elements based on the attributes if it is possible to execute script.

In the following sample, removed script tag and onmouseover attribute from content of the Rich Text Editor.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Prevention of XSS attack sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);

```



```
let rteValue: string = `<div onmouseover='javascript:alert(1) '>Prevention of
Cross Sit Scripting (XSS) </div> <script>alert('hi')</script>`;
let defaultRTE: RichTextEditor = new RichTextEditor({
    value: rteValue
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

It is only applicable to editorMode as HTML.

Custom cross-site scripting

You can also filter the elements and attributes additionally, which cause the XSS attack through [beforeSanitizeHtml](#) event. Return the value from the event argument `helper` function to apply in the editor. To prevent the built-in support and make own cross-site scripting rules, set `cancel` argument to true.

The following sample demonstrates how to filter `script` tag from value.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Custom cross-site scripting sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar,
BeforeSanitizeHtmlArgs } from '@syncfusion/ej2-richtexteditor';
import { detach } from '@syncfusion/ej2-base';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let rteValue: string = `<div>Prevention of Cross Sit Scripting (XSS) </div>
<script>alert('hi')</script>`;
let defaultRTE: RichTextEditor = new RichTextEditor({
    value: rteValue,
    beforeSanitizeHtml: (e: BeforeSanitizeHtmlArgs) => {
        e.helper = (value: string) => {
            e.cancel = true;
            let temp: HTMLElement = document.createElement('div');
            temp.innerHTML = value;
            let scriptTag: HTMLElement = temp.querySelector('script');
            if (scriptTag) {
                detach(scriptTag);
            }
            return temp.innerHTML;
        }
    }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            </div>
        </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Resizable support

This feature allows the editor to be resized dynamically. The users can enable or disable this feature using the `enableResize` property in the Rich Text Editor. If `enableResize` is set to true, the Rich Text Editor component creates grip at the bottom right corner, which allows resizing the component in the diagonal direction. The following sample demonstrates the resizable feature.

Enabling the resizable support

To render the Rich Text Editor in the resizable mode, set the `enableResize` property to true. The above feature is built as module and hence the `Resize` module needs to be included in your application.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Resize } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Resize);
let defaultRTE: RichTextEditor = new RichTextEditor({
    enableResize: true
});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Specifying the Minimum and Maximum width and height for Resize

To have a restricted resizable area for the Rich Text Editor, you need to specify the min-width, max-width, min-height, and max-height CSS properties for the control's wrapper element. By default, the control is capable of resizing upto the current viewport. The `e-richtexteditor` CSS class will be available in the component's wrapper and can be used for applying the above mentioned styles.

```
`css
```

```
.e-richtexteditor {
min-width: 200px;
max-width: 800px;
```

```

min-height: 100px;
max-height: 300px;
}
,

```

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Resize } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Resize);
let defaultRTE: RichTextEditor = new RichTextEditor({
    enableResize: true
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
        </div>
    </div>
</body>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Number and Bullet Format Lists

This feature allows the user to change the appearance of the Numbered and Bulleted lists. Users can also apply different numbering or bullet formats lists such as lowercase greek, upper Alpha, square and circles. You can also customize the style type of the lists to be populated in the dropdown from the toolbar by using the `numberFormatList` and `bulletFormatList` properties in the Rich Text Editor.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    toolbarSettings: {
        items: ['NumberFormatList', 'BulletFormatList']},
});
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
                <li><p>Capable of handling markdown editing.</p></li>
                <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
                <li><p>Provides a fully customizable toolbar.</p></li>
                <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
                <li><p>Supports third-party library integration.</p></li>
                <li><p>Allows preview of modified content before saving
it.</p></li>
                <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Xhtml validation in EJ2 JavaScript Rich text editor control

The editor provides an option to validate the source content of the Rich Text Editor against the XHTML standard using the 'enableXhtml' property. When you enter or modify content in the editor, it continuously checks the XHTML source content and removes elements and attributes that are not valid.

The editor checks the following settings on validation:

Attributes

- Must be specified in lowercase.
- Proper use of quotation marks around the attributes.
- Must be valid attributes for corresponding HTML element.
- All the required attributes must be included in the HTML element.

HTML Elements

- Must be in lowercase.
- All opening tags must be closed.
- Allows only the valid HTML elements.
- Elements must be properly nested.
- All elements must have one root element.
- Should not use inline elements inside the block elements.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Rich Text Editor default sample
 */
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({enableXhtml : true });
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul>
                <li>
                    <p>Provides &#60;IFRAME&#62; and &#60;DIV&#62; modes</p>
                </li>
                <li>
                    <p>Capable of handling markdown editing.</p>
                </li>
                <li>
                    <p>Contains a modular library to load the necessary
functionality on demand.</p>
                </li>
                <li>
                    <p>Provides a fully customizable toolbar.</p>
                </li>
                <li>
                    <p>Provides HTML view to edit the source directly for
developers.</p>
                </li>
                <li>
                    <p>Supports third-party library integration.</p>
                </li>
                <li>
                    <p>Allows preview of modified content before saving
it.</p>
                </li>
                <li>
                    <p>Handles images, hyperlinks, video, hyperlinks,
uploads, etc.</p>
                </li>
            </ul>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Third party integration in EJ2 JavaScript Rich text editor control

The Rich Text Editor can be integrated with third-party to suite the application scenario.

Code-Mirror Integration

Rich Text Editor comes with a basic HTML source editor through view-source property. [Code mirror](#) plugin can be used to highlight the syntax of HTML. CodeMirror plugin for Rich Text Editor makes editing of HTML source code with a pleasant experience.

Import necessary CSS and JS files of CodeMirror to the HTML page.

Required JS files of code mirror.

```

`ts
<script src="scripts/CodeMirror/codemirror.js" type="text/javascript"></script>
<script src="scripts/CodeMirror/javascript.js" type="text/javascript"></script>
<script src="scripts/CodeMirror/css.js" type="text/javascript"></script>
<script src="scripts/CodeMirror/htmlmixed.js" type="text/javascript"></script>
,

```

Required CSS file of code mirror.

```

`ts
<link href="scripts/CodeMirror/codemirror.min.css" rel="stylesheet" />
,

```

Add a custom icon for HTML source editor in the toolbar of Rich Text Editor using template option of [toolbarSettings](#) and define the code mirror plugins, and then pass the Rich Text Editor content as argument in [actionComplete](#) event.

INDEX.TS

```

import { addClass, removeClass, Browser } from '@syncfusion/ej2-base';
import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
import { createElement } from '@syncfusion/ej2-base';
let defaultRTE: RichTextEditor = new RichTextEditor({ toolbarSettings: {
    items: ['SourceCode']},
    showCharCount: true,
    actionComplete: actionCompleteHandler});
defaultRTE.appendTo('#defaultRTE');
let myCodeMirror: any;
function mirrorConversion(e?: any): void {
    let textArea: HTMLElement = defaultRTE.contentModule.getEditPanel() as
HTMLElement;
    let id: string = defaultRTE.getID() + 'mirror-view';

```

```

    let mirrorView: HTMLElement = defaultRTE.element.querySelector('#' + id)
    as HTMLElement;
    let charCount: HTMLElement = defaultRTE.element.querySelector('.e-rte-
character-count') as HTMLElement;
    if (e.targetItem === 'Preview') {
        textArea.style.display = 'block';
        mirrorView.style.display = 'none';
        textArea.innerHTML = myCodeMirror.getValue();
        charCount.style.display = 'block';
    } else {
        if (!mirrorView) {
            mirrorView = createElement('div', { className: 'e-content' });
            mirrorView.id = id;
            textArea.parentNode.appendChild(mirrorView);
        } else {
            mirrorView.innerHTML = '';
        }
        textArea.style.display = 'none';
        mirrorView.style.display = 'block';
        renderCodeMirror(mirrorView, defaultRTE.value);
        charCount.style.display = 'none';
    }
}
function renderCodeMirror(mirrorView: HTMLElement, content: string): void {
    myCodeMirror = CodeMirror(mirrorView, {
        value: content,
        lineNumbers: true,
        mode: 'text/html',
        lineWrapping: true,
    });
}
function actionCompleteHandler(e: any): void {
    if (e.targetItem && (e.targetItem === 'SourceCode' || e.targetItem ===
'Preview')) {
        this.sourceCodeModule.getPanel().style.display = 'none';
        mirrorConversion(e);
    } else {
        setTimeout(() => {
            defaultRTE.toolbarModule.refreshToolbarOverflow(); }, 400);
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<!--CodeMirror references-->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.js"
type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/css/css.js
" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/xml/xml.js
" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/htmlmixed/
htmlmixed.js" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/javascript
/javascript.js" type="text/javascript"></script>
<link
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.min
.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id="defaultRTE">
<p>The Rich Text Editor component is WYSIWYG ("what you see is
what you get") editor that provides the best user experience to create and
update the content.Users can format their content using standard toolbar
commands.</p>
<p><b>Key features:</b></p>
<ul><li><p>Provides <b>IFRAME</b> and <b>#62;DIV</b>
modes</p></li>
<li><p>Capable of handling markdown editing.</p></li>
<li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
<li><p>Provides a fully customizable toolbar.</p></li>
<li><p>Provides HTML view to edit the source directly for
developers.</p></li>
<li><p>Supports third-party library integration.</p></li>
<li><p>Allows preview of modified content before saving
it.</p></li>

```

```

        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
        </ul>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Embed.ly Integration

Rich Text Editor easily integrate with [Embed.ly](#) which is probably the best service when it comes to embed the rich content such as Twitter, Facebook, Instagram and lots of other publishing platform embeds.

`ts

```
<script src="https://cdn.embedly.com/widgets/platform.js" charset="UTF-8"></script>
```

In the following sample, the [Embed.ly](#) class `embedly-card` has been added to `<a>` tag in [actionComplete](#) event.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    placeholder: "Click link icon in toolbar to add the desired link",
    toolbarSettings: {
        items: ['createLink'],
        actionComplete: function(args) {
            if (<String>args.requestType === 'Links') {
                if (args.elements[0].parentNode &&
(<Element>args.elements[0].parentNode).tagName === 'A') {
                    var emberEle=document.createElement('blockquote'); // to add
the link
                    emberEle.setAttribute('class', 'embedly-card'); // to add the
class
                    emberEle.appendChild(args.elements[0].parentElement);
                    emberEle.appendChild(document.createElement('p'));
                    args.range.insertNode(emberEle); // add the link description
to the rte content
                }
            }
        }
    });
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/at.js/1.4.0/css/jquery.atwho.mi
n.css">
  <script src="https://cdn.embedly.com/widgets/platform.js"></script>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
    </div>
  </div>
  <style>
  </style>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Keyboard support in EJ2 JavaScript Rich text editor control

The editor has full keyboard accessibility that includes shortcuts to open and other actions with toolbar items, drop-down lists, and dialogs.

HTML formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with **HTML** [editMode](#).

| Actions | Keyboard shortcuts |

|-----|-----|

| Toolbar focus | Alt + f10 |

| Insert link | Ctrl + k |

| Insert image | Ctrl + Shift + i |

| Insert audio | Ctrl + Shift + a |

| Insert video | Ctrl + Alt + v |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste | Ctrl + v |

| Bold | Ctrl + b |

| Italic | Ctrl + i |

| Underline | Ctrl + u |

| Strikethrough | Ctrl + Shift + s |

| Uppercase | Ctrl + Shift + u |

| Lowercase | Ctrl + Shift + l |

| Superscript | Ctrl + Shift + = |

| Subscript | Ctrl + = |

| Indents | Ctrl +] |

| Outdents | Ctrl + [|

| HTML source | Ctrl + Shift + h |

| Fullscreen | Ctrl + Shift + f |

| Exit Fullscreen | Esc |

| Justify center | Ctrl + e |

| Justify full | Ctrl + j |

| Justify left | Ctrl + l |

| Justify right | Ctrl + r |

| Clear format | Ctrl + Shift + r |

| Ordered list | Ctrl + Shift + o |

| Unordered list | Ctrl + Alt + o |

| Format Painter Copy | Alt + Shift + c |

| Format Painter Paste | Alt + Shift + v |

| Format Painter Escape | Esc |

INDEX.TS

```
import { RichTextEditor, Toolbar, HtmlEditor, Image, QuickToolbar, Link,
FormatPainter } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, HtmlEditor, Image, QuickToolbar, Link,
FormatPainter);
let defaultRTE: RichTextEditor = new RichTextEditor({
  toolbarSettings: {
    items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
'LowerCase', 'UpperCase', '|',
'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
'Outdent', 'Indent', '|',
'CreateLink', 'Image', '|', 'FormatPainter', 'ClearFormat', 'Print',
'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
  }
});
defaultRTE.appendTo('#defaultRTE');
document.onkeyup = function (e) {
  if (e.altKey && e.keyCode === 84 /* t */) {
    // press alt+t to focus the component.
    defaultRTE.focusIn();
  }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/ej2-
popups/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62;; and &#60;DIV&#62;;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Markdown formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with **Markdown** [editMode](#).

| Actions | Keyboard shortcuts |

|-----|-----|

| **Toolbar focus** | **Alt + f10** |

| **Insert link** | **Ctrl + k** |

| **Insert image** | **Ctrl + Shift + i** |

| **Insert table** | **Ctrl + Shift + e** |

| **Undo** | **Ctrl + z** |

| **Redo** | **Ctrl + y** |

| **Copy** | **Ctrl + c** |

| **Cut** | **Ctrl + x** |

| **Paste** | **Ctrl + v** |

| **Bold** | **Ctrl + b** |

| **Italic** | **Ctrl + i** |

| **Strikethrough** | **Ctrl + Shift + s** |

| **Uppercase** | **Ctrl + Shift + u** |

| **Lowercase** | **Ctrl + Shift + l** |

| **Superscript** | **Ctrl + Shift + =** |

| **Subscript** | **Ctrl + =** |

| **Fullscreen** | **Ctrl + Shift + f** |

| **Ordered list** | **Ctrl + Shift + o** |

| **Unordered list** | **Ctrl + Alt + o** |

INDEX.TS

```
import { RichTextEditor, Toolbar, MarkdownEditor } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, MarkdownEditor);
let defaultRTE: RichTextEditor = new RichTextEditor({
  editorMode: 'Markdown',
  toolbarSettings: {
    items: ['Bold', 'Italic', 'StrikeThrough', '|',
            'Formats', 'OrderedList', 'UnorderedList', '|',
            'CreateLink', 'Image', '|', 'Undo', 'Redo']
  }
});
defaultRTE.appendTo('#defaultRTE');
document.onkeyup = function (e) {
  if (e.altKey && e.keyCode === 84 /* t */) {
    // press alt+t to focus the component.
  }
}
```

```

        defaultRTE.focusIn();
    }
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <script
    src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
    <div id="defaultRTE">
      In Rich Text Editor , you click the toolbar buttons to format the words and
      the changes are visible immediately.
      Markdown is not like that. When you format the word in Markdown format, you
      need to add Markdown syntax to the word to indicate which words
      and phrases should look different from each other
      Rich Text Editor supports markdown editing when the editorMode set as
      **markdown** and using both *keyboard interaction* and *toolbar action*, you
      can apply the formatting to text.
      We can add our own custom formation syntax for the Markdown formation,
      [sample link] (https://ej2.syncfusion.com/home/).
      The third-party library <b>Marked</b> is used in this sample to convert
      markdown into HTML content
    </div>
  </div>

```

```

</div>
<style>
    .e-richtexteditor textarea.e-content {
        float: left;
        border-right: 1px solid rgba(0, 0, 0, 0.12);
    }
    .e-richtexteditor .e-rte-content .e-content {
        min-height: 150px;
    }

    .e-richtexteditor .e-rte-content {
        overflow: hidden;
    }

    .e-icon-btn.e-active .e-md-preview::before {
        content: '\e350';
    }

    .e-icon-btn .e-md-preview::before {
        content: '\e345';
    }

    .e-rte-content .e-content {
        float: right;
        width: 50%;
        overflow: auto;
        height: inherit;
        padding: 8px;
        height: 100%;
    }

    .e-rte-content .e-content.e-pre-source {
        width: 100%;
    }

    .highcontrast .e-richtexteditor textarea.e-content {
        border-right: 1px solid #fff;
    }

    .sb-header {
        z-index: 100;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom key config

Customize the key config for the keyboard interaction of Rich Text Editor, using the [keyConfig](#) property.

In the following sample, customize the cut, copy, and paste toolbar action with **Ctrl+1**, **Ctrl+2**, and **Ctrl+3**, respectively.

INDEX.TS

```
import { RichTextEditor, Toolbar, HtmlEditor } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, HtmlEditor);
let defaultRTE: RichTextEditor = new RichTextEditor({
  toolbarSettings: {
    items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
      'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
      'LowerCase', 'UpperCase', '|',
      'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
      'Outdent', 'Indent', '|',
      'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
      'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
  },
  keyConfig: {
    'copy': 'ctrl+1',
    'cut': 'ctrl+2',
    'paste': 'ctrl+3'
  }
});
defaultRTE.appendTo('#defaultRTE');
document.onkeyup = function (e) {
  if (e.altKey && e.keyCode === 84 /* t */) {
    // press alt+t to focus the component.
    defaultRTE.focusIn();
  }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul><li><p>Provides &#60;IFRAME&#62;; and &#60;DIV&#62;;
modes</p></li>
            <li><p>Capable of handling markdown editing.</p></li>
            <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
            <li><p>Provides a fully customizable toolbar.</p></li>
            <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
            <li><p>Supports third-party library integration.</p></li>
            <li><p>Allows preview of modified content before saving
it.</p></li>
            <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
            </ul>
        </div>
    </div>
    <style>
        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
            content: "\e76e";
        }

        .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
            content: "\e726";
        }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Globalization](#)
- [Accessibility](#)
- [How to customize the toolbar items shortcut key](#)
- [How to customize the saving operation](#)

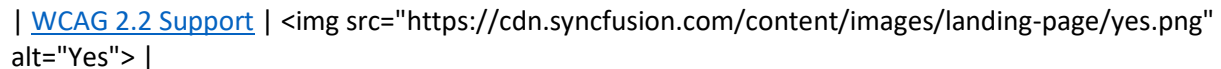
Accessibility in EJ2 JavaScript Rich text editor control

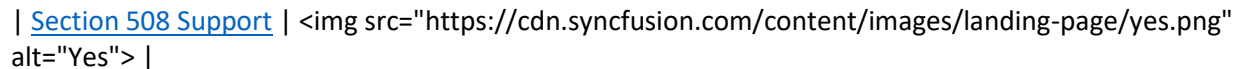
The Rich Text Editor component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties. This component is characterized by complete ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation

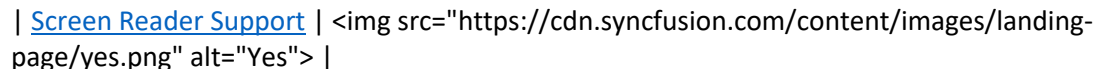
The accessibility compliance for the Rich Text Editor component is outlined below.

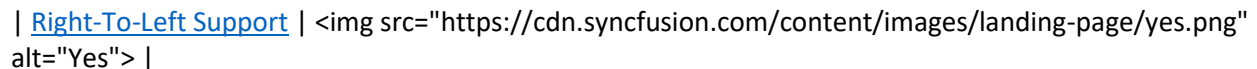
| [Accessibility Criteria](#) | [Compatibility](#) |

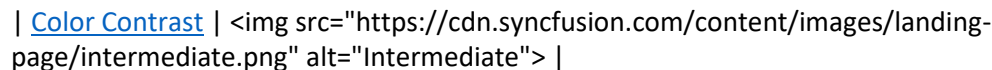
| -- | -- |

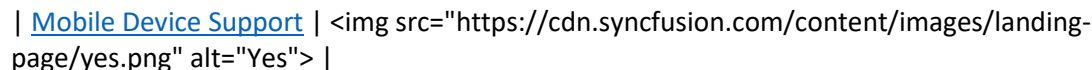
| [WCAG 2.2 Support](#) |  alt="Yes" > |


| [Section 508 Support](#) |  alt="Yes" > |

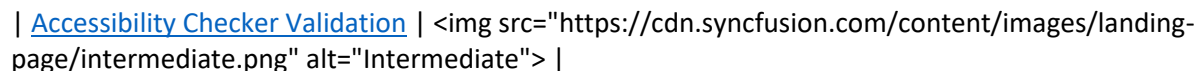
| [Screen Reader Support](#) |  alt="Yes" > |

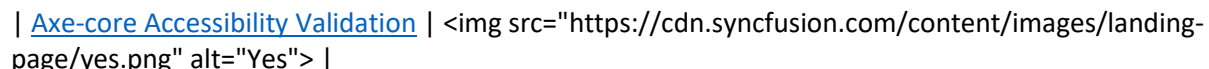
| [Right-To-Left Support](#) |  alt="Yes" > |

| [Color Contrast](#) |  alt="Intermediate" > |

| [Mobile Device Support](#) |  alt="Yes" > |

| [Keyboard Navigation Support](#) |  alt="Yes" > |

| [Accessibility Checker Validation](#) |  alt="Intermediate" > |

| [Axe-core Accessibility Validation](#) |  alt="Yes" > |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

ARIA attributes

- The toolbar of Rich Text Editor, assigned the role of **Toolbar** and has the following list of ARIA attribute.

| Property | Functionalities |

| --- | --- |

| role="toolbar" | This attribute added to the ToolBar element describes the actual role of the element. |

| aria-orientation | Indicates the ToolBar orientation. Default value is **horizontal**. |

| aria-haspopup | Indicates the popup mode of the Toolbar. Default value is false. When popup mode is enabled, attribute value has to be changed to **true**. | |

| aria-disabled | Indicates the disabled state of the ToolBar. |

| aria-owns | Identifies an element to define a visual, functional, or contextual parent/child relationship between DOM elements when the DOM hierarchy cannot represent the relationship. In the Rich Text Editor, the attribute contains the ID of the Rich Text Editor to indicate the popup as a child element. |

For further details of Toolbar ARIA attributes, refer the [accessibility of Toolbar](#) documentation.

The Rich Text Editor element is assigned the role of **application**.

| Property | Functionalities |

| --- | --- |

| role="application" | This attribute added to the Rich Text Editor element describes the actual role of the element. |

| aria-disabled | Indicates the disabled state of the ToolBar. |

INDEX.TS

```
import { RichTextEditor, Toolbar, HtmlEditor } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, HtmlEditor);
let defaultRTE: RichTextEditor = new RichTextEditor({
  toolbarSettings: {
    items: ['Bold', 'Italic', 'Underline', 'StrikeThrough',
      'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
      'LowerCase', 'UpperCase', '|',
      'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
      'Outdent', 'Indent', '|',
      'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
```



```

        'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
    });
    defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
      <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
      <li><p>Provides a fully customizable toolbar.</p></li>
      <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
      <li><p>Supports third-party library integration.</p></li>

```

```

        <li><p>Allows preview of modified content before saving
it.</p></li>
        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
    </ul>
</div>
</div>
<style>
    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
        content: "\e76e";
    }

    .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
        content: "\e726";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Keyboard interaction

The Rich Text Editor component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Rich Text Editor component.

HTML formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with HTML edit mode.

| Actions | Keyboard shortcuts |

|-----|-----|

| Toolbar focus | Alt + f10 |

| Insert link | Ctrl + k |

| Insert image | Ctrl + Shift + i |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste | Ctrl + v |

| Bold | Ctrl + b |

| Italic | Ctrl + i |

Underline	Ctrl + u
Strikethrough	Ctrl + Shift + s
Uppercase	Ctrl + Shift + u
Lowercase	Ctrl + Shift + l
Superscript	Ctrl + Shift + =
Subscript	Ctrl + =
Indents	Ctrl +]
Outdents	Ctrl + [
HTML source	Ctrl + Shift + h
Fullscreen	Ctrl + Shift + f
Exit Fullscreen	Esc
Justify center	Ctrl + e
Justify full	Ctrl + j
Justify left	Ctrl + l
Justify right	Ctrl + r
Clear format	Ctrl + Shift + r
Ordered list	Ctrl + Shift + o
Unordered list	Ctrl + Alt + o
Format Painter Copy	Alt + Shift + c
Format Painter Paste	Alt + Shift + v
Format Painter Escape	Esc

Markdown formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with Markdown edit mode

| Actions | Keyboard shortcuts |

|-----|-----|

Toolbar focus	Alt + f10
Insert link	Ctrl + k
Insert image	Ctrl + Shift + i
Insert table	Ctrl + Shift + e
Undo	Ctrl + z
Redo	Ctrl + y
Copy	Ctrl + c

- | Cut | Ctrl + x |
- | Paste | Ctrl + v |
- | Bold | Ctrl + b |
- | Italic | Ctrl + i |
- | Strikethrough | Ctrl + Shift + s |
- | Uppercase | Ctrl + Shift + u |
- | Lowercase | Ctrl + Shift + l |
- | Superscript | Ctrl + Shift + = |
- | Subscript | Ctrl + = |
- | Fullscreen | Ctrl + Shift + f |
- | Ordered list | Ctrl + Shift + o |
- | Unordered list | Ctrl + Alt + o |

Ensuring accessibility

The Rich Text Editor component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Rich Text Editor component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Rich Text Editor component with accessibility tools.

See also

- [Accessibility in Syncfusion JavaScript components](#)

How To

Add google font in EJ2 JavaScript Rich text editor control

To use web fonts in Rich Text Editor, it is not needed for the web fonts to be present in local machine.

To add the web fonts to Rich Text Editor, we need to refer the web font links and add the font names in the [fontFamily](#) property.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
  fontFamily: {
    items: [
      {text: "Segoe UI", value: "Segoe UI", class: "e-segoe-ui",
command: "Font", subCommand: "FontName"},
      {text: "Roboto", value: "Roboto", command: "Font", subCommand:
"FontName"},
      // here font is added
    ]
  }
});
```

```

        {text: "Great vibes", value: "Great Vibes,cursive", command:
"Font", subCommand: "FontName"},// here font is added
        {text: "Noto Sans", value: "Noto Sans", command: "Font",
subCommand: "FontName"},
        {text: "Impact", value: "Impact,Charcoal,sans-serif", class: "e-
impact", command: "Font", subCommand: "FontName"},
        {text: "Tahoma", value: "Tahoma,Geneva,sans-serif", class: "e-
tahoma", command: "Font", subCommand: "FontName"},
    ]
},
toolbarSettings: {
    items: ['Bold', 'Italic', 'Underline', 'StrikeThrough','|',
'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
'LowerCase', 'UpperCase', '|',
'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
'Outdent', 'Indent', '|',
'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
},
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Roboto">
  <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Great+Vibes">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul>
                <li>
                    <p>Provides <b>IFRAME</b> and <b>DIV</b> modes</p>
                </li>
                <li>
                    <p>Capable of handling markdown editing.</p>
                </li>
                <li>
                    <p>Contains a modular library to load the necessary
functionality on demand.</p>
                </li>
                <li>
                    <p>Provides a fully customizable toolbar.</p>
                </li>
            </ul>
        </div>
    </div>
    <style>
    </style>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The below font style links are referred in the page.

`ts

<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto">

<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Great+Vibes">

,

In the above sample, you can see that we have added two Google web fonts (Roboto and Great vibes) to Rich Text Editor.

Default font in EJ2 JavaScript Rich text editor control

By using [default](#) property, you can change the default font-family of the Rich Text Editor. To change the font-family of the Rich Text Editor content while loading, we need to give the font-family in the style section with the help of [cssClass](#) property.

INDEX.TS

```
import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
  fontFamily: {
    default: "Noto Sans", // to define default font-family
    items: [
      {text: "Segoe UI", value: "Segoe UI", class: "e-segoe-ui",
command: "Font", subCommand: "FontName"},
      {text: "Noto Sans", value: "Noto Sans", command: "Font",
subCommand: "FontName"},
      {text: "Impact", value: "Impact,Charcoal,sans-serif", class: "e-
impact", command: "Font", subCommand: "FontName"},
      {text: "Tahoma", value: "Tahoma,Geneva,sans-serif", class: "e-
tahoma", command: "Font", subCommand: "FontName"},
    ]
  },
  toolbarSettings: {
    items: ['Bold', 'Italic', 'Underline', 'StrikeThrough', '|',
'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
'LowerCase', 'UpperCase', '|',
'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
'Outdent', 'Indent', '|',
'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
  },
  cssClass: "customClass"
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="defaultRTE">
            <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
            <p><b>Key features:</b></p>
            <ul>
                <li>
                    <p>Provides &#60;IFRAME&#62; and &#60;DIV&#62; modes</p>
                </li>
                <li>
                    <p>Capable of handling markdown editing.</p>
                </li>
                <li>
                    <p>Contains a modular library to load the necessary
functionality on demand.</p>
                </li>
            </ul>
        </div>
    </div>
    <style>
        .customClass .e-rte-content .e-content {
            /* to get the desired font on intially*/
            font-family: "Noto Sans";
        }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


File size in EJ2 JavaScript Rich text editor control

By using the Rich text editor's **imageUploading** event, you can get the image size before uploading and restrict the image to upload, when the given image size is greater than the allowed size.

In the following, we have validated the image size before uploading and determined whether the image has been uploaded or not.

INDEX.TS

```
import { RichTextEditor, Toolbar, Image, Link, HtmlEditor, QuickToolbar,
NodeSelection } from '@syncfusion/ej2-richtexteditor';
import { UploadingEventArgs } from '@syncfusion/ej2-inputs';
RichTextEditor.Inject(Toolbar, Image, Link, HtmlEditor, QuickToolbar );
let defaultRTE: RichTextEditor = new RichTextEditor({
    height: 400,
    toolbarSettings: {
        items: ['Undo', 'Redo', '|',
                'Bold', 'Italic', 'Underline', 'StrikeThrough', '|',
                'FontName', 'FontSize', 'FontColor', 'BackgroundColor', '|',
                'SubScript', 'SuperScript', '|',
                'LowerCase', 'UpperCase', '|',
                'Formats', 'Alignments', '|', 'OrderedList',
                'UnorderedList', '|',
                'Indent', 'Outdent', '|', 'Image', '|', 'SourceCode',
                '|', 'ClearFormat', 'Print']
    },
    insertImageSettings: {
        saveUrl:
            "https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save",
        path: "../Images/"
    },
    imageUploading: onImageUploading
});
defaultRTE.appendTo("#defaultRTE");
function onImageUploading(args: UploadingEventArgs): void {
    console.log("file is uploading");
    let imgSize = 500000;
    let sizeInBytes: number = args.fileData.size;
    if ( imgSize < sizeInBytes ) {
        args.cancel = true;
    }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Rich Text Editor</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
        <div id="defaultRTE">
            </div>
        </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Shortcut key in EJ2 JavaScript Rich text editor control

It can be achieved by using [formatter](#) property. We need to create `customformatterModel` to configure the `keyConfig` using `IHtmlFormatterModel` class and assign the same to the formatter property. Here, `ctrl+q` is configured to open the `Insert Hyperlink` dialog.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, Count, IHtmlFormatterModel,
HTMLFormatter, HtmlEditor, QuickToolbar } from '@syncfusion/ej2-
richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
let customHTMLModel: IHtmlFormatterModel = { // formatter is used to
configure the custom key
    keyConfig: {
        'insert-link': 'ctrl+q', // configure the desired key
    }
};

```

```

    }
  };
  let defaultRTE: RichTextEditor = new RichTextEditor({
    formatter: new HTMLFormatter(customHTMLModel), // to configure custom
    key
  });
  defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Roboto">
  <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Great+Vibes">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul>
        <li>

```

```

        <p>Provides <b>IFRAME</b> and <b>DIV</b> modes</p>
    </li>
    <li>
        <p>Capable of handling markdown editing.</p>
    </li>
    <li>
        <p>Contains a modular library to load the necessary
functionality on demand.</p>
    </li>
    <li>
        <p>Provides a fully customizable toolbar.</p>
    </li>
</ul>
</div>
</div>
<style>
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

We need to import `IHtmlFormatterModel` and `HTMLFormatter` to configure the shortcut key.

Save in `EJ2 JavaScript Rich text editor control`

To achieve this, we need to bind the `keydown` event to the Rich Text Editor content and capture the `ctrl + s` key press using its `keyCode`.

In the `keydown` event handler, the `updateValue` method is called to update the `value` property and then we can save the content in the required database using the same.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({});
defaultRTE.appendTo('#defaultRTE');
defaultRTE.contentModule.getDocument().addEventListener("keydown", function(e
: any):void{
    if(e.key === 's' && e.ctrlKey===true){
        e.preventDefault(); // to prevent default ctrl+s action
        defaultRTE.updateValue(); // to update the value after editing
        let value: any= defaultRTE.value; // you can get the RTE content to
save in the desired database
    }
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Roboto">
  <link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Great+Vibes">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul>
        <li>
          <p>Provides <b>IFRAME</b> and <b>DIV</b> modes</p>
        </li>
        <li>
          <p>Capable of handling markdown editing.</p>
        </li>
        <li>
          <p>Contains a modular library to load the necessary
functionality on demand.</p>
        </li>
        <li>
          <p>Provides a fully customizable toolbar.</p>
        </li>
      </ul>
    </div>
  </div>

```

```

        </li>
      </ul>
    </div>
  </div>
  <style>
  </style>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Placeholder in EJ2 JavaScript Rich text editor control

By using `e-rte-placeholder` class, you can customize the placeholder style.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Count } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar,
Count);
let defaultRTE: RichTextEditor = new RichTextEditor({ placeholder: 'Type
Something' });
defaultRTE.appendTo('#defaultRTE');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
        <div id="defaultRTE">
        </div>
    </div>
    <style>
        .e-richtexteditor .e-rte-placeholder {
            /* placeholder style */
            font-family: monospace;
            color: deeppink;
        }
    </style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cursor in EJ2 JavaScript Rich text editor control

This can be achieved by using `setRange` method in the Rich Text Editor using `NodeSelection` instance. In this below sample, we have passed the text node (specific location in Rich Text Editor content) in `setStart` method and passed the range in `setRange` method of Rich Text Editor.

INDEX.TS

```

import { addClass, removeClass, Browser, isNullOrUndefined } from
'@syncfusion/ej2-base';
import { RichTextEditor, NodeSelection, Toolbar, Link, Image, Count,
HtmlEditor,
    QuickToolbar, ActionBeginEventArgs, IToolsItems } from '@syncfusion/ej2-
richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor,
QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    placeholder: 'Type Something',
});
defaultRTE.appendTo('#defaultRTE');
document.getElementById('btn').onclick = function() {
    let element: Element =
defaultRTE.contentModule.getDocument().getElementById("key");

```

```
let selectioncursor: NodeSelection = new NodeSelection();
let range: Range = document.createRange();
range.setStart(element, 1); // to set the range
selectioncursor.setRange(document, range); // to set the cursor
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p id="key"><b>Key features:</b></p>
      <ul>
        <li>
          <p>Provides &#60;IFRAME&#62; and &#60;DIV&#62; modes</p>
        </li>
        <li>
          <p>Capable of handling markdown editing.</p>
        </li>
      </ul>
    </div>
  </div>
```



```

        </li>
        <li>
            <p>Contains a modular library to load the necessary
functionality on demand.</p>
        </li>
        <li>
            <p>Provides a fully customizable toolbar.</p>
        </li>
    </ul>
</div>
<input id="btn" type="button" class="e-btn" style="margin-top:40px"
value="Set cursor point">
</div>
<style>
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Rename images in server in EJ2 JavaScript Rich text editor control

By using the [insertImageSettings](#) property, you can specify the server handler to upload the selected images. Then you can bind the [imageUploadSuccess](#) event, to receive the modified file name from the server and update it in the Rich Text Editor's insert image dialog.

`html

```
<div id='defaultRTE'>
```

```
<p>The Rich Text Editor is WYSIWYG ("what you see is what you get") editor useful to create and edit
content, and return the valid <a href="https://ej2.syncfusion.com/home/" target="blank">HTML
markup</a> or <a href="https://ej2.syncfusion.com/home/" target="blank">markdown</a> of the
content</p>
```

```
<p><b>Key features:</b></p>
```

```
<ul>
```

```
<li>
```

```
<p>Provides <IFRAME> and <DIV> modes</p>
```

```
</li>
```

```
<li>
```

```
<p>Capable of handling markdown editing.</p>
```

```
</li>
```

```
<li>
```

```
<p>Contains a modular library to load the necessary functionality on demand.</p>
```

```

</li>
</ul>
</div>
`
`ts
import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar } from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
toolbarSettings: {
items: ['Bold', 'Italic', 'Underline', 'StrikeThrough', '|',
'FontName', 'FontSize', 'FontColor', 'BackgroundColor',
'LowerCase', 'UpperCase', '|',
'Formats', 'Alignments', 'OrderedList', 'UnorderedList',
'Outdent', 'Indent', '|',
'CreateLink', 'Image', '|', 'ClearFormat', 'Print',
'SourceCode', 'FullScreen', '|', 'Undo', 'Redo']
},
insertImageSettings: {
saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Rename",
path: "../Images/"
},
imageUploadSuccess: onImageUploadSuccess
});
defaultRTE.appendTo('#defaultRTE');
function onImageUploadSuccess (args: any) {
alert("Get the new file name here");
if (args.e.currentTarget.getResponseHeader('name') != null) {
args.file.name = args.e.currentTarget.getResponseHeader('name');
let filename: any = document.querySelectorAll(".e-file-name")[0];
filename.innerHTML = args.file.name.replace(document.querySelectorAll(".e-file-type")[0].innerHTML, "");
filename.title = args.file.name;
}
}

```

```
}  
,
```

To configure server-side handler, refer the below code.

```
`c#  
int x = 0;  
string file;  
[AcceptVerbs("Post")]  
public void Rename()  
{  
    try  
    {  
        var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];  
        imageFile = httpPostedFile.FileName;  
        if (httpPostedFile != null)  
        {  
            var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Images");  
            if (!Directory.Exists(fileSave))  
            {  
                Directory.CreateDirectory(fileSave);  
            }  
            var fileName = Path.GetFileName(httpPostedFile.FileName);  
            var fileSavePath = Path.Combine(fileSave, fileName);  
            while (System.IO.File.Exists(fileSavePath))  
            {  
                imageFile = "rtImage" + x + "-" + fileName;  
                fileSavePath = Path.Combine(fileSave, imageFile);  
                x++;  
            }  
            if (!System.IO.File.Exists(fileSavePath))  
            {  
                httpPostedFile.SaveAs(fileSavePath);  
                HttpResponseMessage Response = System.Web.HttpContext.Current.Response;  
                Response.Clear();
```

```

Response.Headers.Add("name", imageFile);
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
}
}
catch (Exception e)
{
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
    Response.Clear();
    Response.ContentType = "application/json; charset=utf-8";
    Response.StatusCode = 204;
    Response.Status = "204 No Content";
    Response.StatusDescription = e.Message;
    Response.End();
}
}
,

```

Format code block in EJ2 JavaScript Rich text editor control

You can configure code block formatting as a separate toolbar button by adding the **InsertCode** keyword within the [toolbarSettings](#) items property.

The InsertCode button has a toggle state to apply code block formatting to the editor and remove code block formatting from the editor.

The following sample demonstrates how to config the InsertCode button in toolbar and set the background color to “pre” tag for highlighting the code block.

INDEX.TS

```

import { RichTextEditor, Toolbar, Link, Image, HtmlEditor, QuickToolbar }
from '@syncfusion/ej2-richtexteditor';
RichTextEditor.Inject(Toolbar, Link, Image, HtmlEditor, QuickToolbar);
let defaultRTE: RichTextEditor = new RichTextEditor({
    toolbarSettings: {
        items: ['InsertCode']
    }
});
defaultRTE.appendTo('#defaultRTE');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Rich Text Editor</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
richtexteditor/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <script
src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
    <div id="defaultRTE">
      <p>The Rich Text Editor is WYSIWYG ("what you see is what you
get") editor useful to create and edit content, and return the valid <a
href="https://ej2.syncfusion.com/home/" target="_blank">HTML markup</a> or
<a href="https://ej2.syncfusion.com/home/" target="_blank">markdown</a> of
the content</p>
      <p><b>Key features:</b></p>
      <ul><li><p>Provides &#60;IFRAME&#62; and &#60;DIV&#62;
modes</p></li>
      <li><p>Capable of handling markdown editing.</p></li>
      <li><p>Contains a modular library to load the necessary
functionality on demand.</p></li>
      <li><p>Provides a fully customizable toolbar.</p></li>
      <li><p>Provides HTML view to edit the source directly for
developers.</p></li>
      <li><p>Supports third-party library integration.</p></li>
      <li><p>Allows preview of modified content before saving
it.</p></li>
    </div>
  </div>

```

```

        <li><p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p></li>
        </ul>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

File attachment in EJ2 JavaScript Rich text editor control

The Rich Text Editor allows you to attach a file based on the file upload. You can attach your files using the file upload or drag-and-drop from your local path. When the file upload gets success, the attachment link inserts into the content.

In the below sample, configure the saveUrl and path properties to achieve file attachments.

1. saveUrl: Provides service URL to save the files.
2. path: Specifies the location to store the image.

The following sample illustrates how to attach a file in the Rich Text Editor.

```
`ts
```

```
import { RichTextEditor, Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar, NodeSelection } from
 '@syncfusion/ej2-richtexteditor';
```

```
RichTextEditor.Inject(Toolbar, Link, Image, Count, HtmlEditor, QuickToolbar, NodeSelection);
```

```
import { Uploader } from '@syncfusion/ej2-inputs';
```

```
let selection: NodeSelection = new NodeSelection();
```

```
let range: Range;
```

```
let saveSelection: NodeSelection;
```

```
let defaultRTE: RichTextEditor = new RichTextEditor({
```

```
    insertImageSettings: {
```

```
        saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Save",
```

```
        path: "../Files/"
```

```
    }
```

```
});
```

```
defaultRTE.appendTo("#defaultRTE");
```

```
let uploadObj: Uploader = new Uploader({
```

```
    asyncSettings: {
```

```

saveUrl: "[SERVICEHOSTEDPATH]/api/uploadbox/Save",
},
dropArea: defaultRTE.inputElement,
success: onUploadSuccess
});
uploadObj.appendTo("#fileupload");
function onUploadSuccess(args: any): void {
(defaultRTE.contentModule.getEditPanel() as HTMLElement).focus();
range = selection.getRange(document);
saveSelection = selection.save(range, document);
let fileUrl: any =
document.URL + defaultRTE.insertImageSettings.path + args.file.name;
if (defaultRTE.formatter.getUndoRedoStack().length === 0) {
defaultRTE.formatter.saveData();
}
saveSelection.restore();
defaultRTE.executeCommand('createLink', { url: fileUrl, text: fileUrl, selection: saveSelection });
defaultRTE.formatter.saveData();
defaultRTE.formatter.enableUndo(defaultRTE);
this.clearAll();
}
`

```

To configure server-side handler, refer the below code.

```

`c#
int x = 0;
string file;
[AcceptVerbs("Post")]
public void Save()
{
try
{
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
file = httpPostedFile.FileName;

```

```
if (httpPostedFile != null)
{
    Console.WriteLine(System.Web.HttpContext.Current.Server.MapPath("~/Files"));
    var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Files");
    if (!Directory.Exists(fileSave))
    {
        Directory.CreateDirectory(fileSave);
    }
    var fileName = Path.GetFileName(httpPostedFile.FileName);
    var fileSavePath = Path.Combine(fileSave, fileName);
    while (System.IO.File.Exists(fileSavePath))
    {
        file = "rte" + x + "-" + fileName;
        fileSavePath = Path.Combine(fileSave, file);
        x++;
    }
    if (!System.IO.File.Exists(fileSavePath))
    {
        httpPostedFile.SaveAs(fileSavePath);
        HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
        Response.Clear();
        Response.Headers.Add("name", file);
        Response.ContentType = "application/json; charset=utf-8";
        Response.StatusDescription = "File uploaded succesfully";
        Response.Headers.Add("url", fileSavePath);
        Response.End();
    }
}
catch (Exception e)
{
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
    Response.Clear();
```



```
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 204;
Response.Status = "204 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
`
```

Schedule

Getting started in EJ2 JavaScript Schedule control

This section briefly explains how to create the [JavaScript Scheduler](#) component and configure its available functionalities in a JavaScript application.

Dependencies

The following list of dependencies are required to use the Scheduler component in your application.

```
`javascript
|-- @syncfusion/ej2-schedule
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-excel-export
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-navigations
`
```

Setup for local environment

Refer the following steps for setup your local environment.

Step 1: Create a root folder `myapp` for your application.

Step 2: Create `myapp/resources` folder to store local scripts and styles files.

Step 3: Create `myapp/index.js` and `myapp/index.html` files for initializing Essential JS 2 Scheduler control.

Adding Syncfusion resources

The Essential JS 2 Scheduler control can be initialized by using either of the following ways.

- Using local scripts and styles.
- Using CDN links for scripts and styles.

Using local scripts and styles

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the Scheduler and its dependency scripts and style files into the `resources/scripts` and `resources/styles` folder respectively.

Refer the below location from where the Scheduler's script and styles can be referenced.

Syntax:

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js**

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

Example:

Script: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-schedule/dist/global/ej2-schedule.min.js**

Styles: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-schedule/styles/material.css**

After copying the files, you can refer the Scheduler's scripts and styles into the `index.html` file.

The below html code example shows the dependency of Scheduler.

```
<!DOCTYPE html>
<html xmlns="https://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Scheduler</title>
<!-- Essential JS 2 Scheduler's dependent material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/calendars/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
```

```

<link href="resources/inputs/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/splitbuttons/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/lists/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/navigations/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Scheduler's material theme -->
<link href="resources/schedule/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Scheduler's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calendars.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-splitbuttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-excel-export.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Scheduler's global script -->
<script src="resources/scripts/ej2-schedule.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>

```

Using CDN links for scripts and styles

Using CDN link, you can directly refer the Scheduler's script and styles into the `index.html` file.

Refer the Scheduler's CDN links as given below.

Syntax:

Script:

https://cdn.syncfusion.com/ej2/{RELEASEVERSION}/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js

Styles: <https://cdn.syncfusion.com/ej2/{RELEASEVERSION}/{PACKAGENAME}/styles/material.css>

Example:

Script: <https://cdn.syncfusion.com/ej2/20.3.56/ej2-schedule/dist/global/ej2-schedule.min.js>

Styles: <https://cdn.syncfusion.com/ej2/20.3.56/ej2-schedule/styles/material.css>

The below html code example shows the dependency of Scheduler with `ej2-schedule.min.js`.

`html

```
<!DOCTYPE html>
<html xmlns="https://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Scheduler</title>
<!-- Essential JS 2 Scheduler's dependent material theme -->
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-base/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-buttons/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-calendars/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-dropdowns/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-inputs/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-splitbuttons/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-lists/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-popups/styles/material.css"
rel="stylesheet" type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-navigations/styles/material.css"
rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Scheduler's material theme -->
```

```
<link href="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-schedule/styles/material.css"
rel="stylesheet" type="text/css"/>

<!-- Essential JS 2 Scheduler's dependent script -->

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-inputs/dist/global/ej2-
inputs.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-calendars/dist/global/ej2-
calendars.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-file-utils/dist/global/ej2-file-
utils.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-buttons/dist/global/ej2-
buttons.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-navigations/dist/global/ej2-
navigations.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-popups/dist/global/ej2-
popups.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-dropdowns/dist/global/ej2-
dropdowns.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-splitbuttons/dist/global/ej2-
splitbuttons.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-compression/dist/global/ej2-
compression.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-excel-export/dist/global/ej2-excel-
export.min.js" type="text/javascript"></script>

<!-- Essential JS 2 Scheduler's global script -->

<script src="https://cdn.syncfusion.com/ej2/{RELEASE_VERSION}/ej2-schedule/dist/global/ej2-
schedule.min.js" type="text/javascript"></script>

</head>

<body>

<!-- Add the HTML <div> element for scheduler -->

<div id="Schedule"></div>

<script src="index.js" type="text/javascript"></script>

</body>
```

```
</html>
```

```
,
```

Place the following Scheduler code in the `index.js` file.

```
`javascript
ej.schedule.Schedule.Inject(ej.schedule.Day, ej.schedule.Week, ej.schedule.WorkWeek,
ej.schedule.Month, ej.schedule.Agenda);
var scheduleObj = new ej.schedule.Schedule();
scheduleObj.appendTo('#Schedule');
,
```

Initialize the Scheduler

Now, you can start adding Scheduler control in the application. For getting started, add a `div` element for Scheduler control in `index.html`. Then refer the `index.js` file into the `index.html` file.

In this document context we are going to use `ej2.min.js` which includes all the Essential JS 2 components and its dependent scripts.

```
`html
<!DOCTYPE html>
<html xmlns="https://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Scheduler</title>
<!-- Essential JS 2 Scheduler's dependent material theme -->
<link href="https://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-calendars/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
```

```

<link href="https://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Scheduler's material theme -->
<link href="https://cdn.syncfusion.com/ej2/ej2-schedule/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 all script -->
<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for scheduler -->
<div id="Schedule"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
`

```

Place the following Scheduler code in the `index.js` file.

```

`javascript
var scheduleObj = new ej.schedule.Schedule();
scheduleObj.appendTo('#Schedule');
`

```

Populating appointments

To populate the empty Scheduler with appointments, define either the local JSON data or remote data through the `dataSource` property available within the `eventSettings` option. To define any appointments, start and end time fields are mandatory. In the following example, you can see the appointment defined with default fields such as Id, Subject, StartTime and EndTime.

```

`ts
var scheduleObj = new ej.schedule.Schedule({
height: '550px',
selectedDate: new Date(2018, 1, 15),
eventSettings: {
dataSource: [{
id: 1,
subject: 'Meeting',
startTime: new Date(2018, 1, 15, 10, 0),
endTime: new Date(2018, 1, 15, 12, 30)
}
]
}
}
)
`

```

```

    }}
  }
});
scheduleObj.appendTo('#Schedule');
`

```

You can also provide different names to these default fields, for which the custom names of those fields must be mapped appropriately within `fields` property as shown below.

```

`ts
var data = [{
  Id: 2,
  EventName: 'Meeting',
  StartTime: new Date(2018, 1, 15, 10, 0),
  EndTime: new Date(2018, 1, 15, 12, 30),
  IsAllDay: false,
  Status: 'Completed',
  Priority: 'High'
}];
var scheduleObj = new ej.schedule.Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: {
    dataSource: data,
    fields: {
      id: 'Id',
      subject: { name: 'EventName' },
      isAllDay: { name: 'IsAllDay' },
      startTime: { name: 'StartTime' },
      endTime: { name: 'EndTime' },
    }
  }
});
scheduleObj.appendTo('#Schedule');
`

```


The other fields available in Scheduler can be referred from [here](#).

Setting date

Scheduler usually displays the system date as its current date. To change the current date of Scheduler with specific date, define the `selectedDate` property.

```
`ts
var scheduleObj = new ej.schedule.Schedule({
height: '550px',
selectedDate: new Date(2018, 1, 15)
});
scheduleObj.appendTo('#Schedule');
`
```

Setting view

Scheduler displays `Week` view by default. To change the current view, define the applicable view name to the `currentView` property. The applicable view names are,

- Day
- Week
- Workweek
- Month
- Year
- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear

```
`ts
var scheduleObj = new ej.schedule.Schedule({
height: '550px',
selectedDate: new Date(2018, 1, 15),
currentView: 'Month'
});
scheduleObj.appendTo('#Schedule');
`
```

Individual view customization

Each individual Scheduler views can be customized with its own options such as setting different start and end hour on Week and Work Week views, whereas hiding the weekend days on Month view alone.

This can be achieved by defining views property to accept the array of object type, where each object depicts the individual view customization.

Now, run the application in the browser using the following command.

```
,
npm start
,
```

The output will display the Scheduler with the specified view configuration.

INDEX.JS

```
ej.schedule.Schedule.Inject(ej.schedule.Week, ej.schedule.WorkWeek,
ej.schedule.Month);
var scheduleObj = new ej.schedule.Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: [{ option: 'Week', startHour: '07:00', endHour: '15:00'},
          { option: 'WorkWeek', startHour: '10:00', endHour: '18:00'},
          { option: 'Month', showWeekend: false }],
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can also explore our [JavaScript Scheduler example](#) that shows how to use the toolbar buttons to play with Scheduler functionalities.

Module injection in EJ2 JavaScript Schedule control

The crucial step on creating a Scheduler with required views, is to import and inject the required modules. The modules that are available on Scheduler to work with its related functionalities are as follows.

- **Day** - Inject this module to work with day view.
- **Week** - Inject this module to work with week view.
- **WorkWeek** - Inject this module to work with work week view.
- **Month** - Inject this module to work with month view.
- **Agenda** - Inject this module to work with agenda view.
- **MonthAgenda** - Inject this module to work with month agenda view.
- **TimelineViews** - Inject this module to work with timeline day, timeline week, and timeline work week views.
- **TimelineMonth** - Inject this module to work with timeline month view.
- **DragAndDrop** - Inject this module to allow drag and drop of appointments on Scheduler.
- **Resize** - Inject this module for enabling the resize functionality of appointments on Scheduler.
- **ExcelExport** - Inject this module for exporting the Scheduler events data as excel file format.
- **ICalendarExport** - Inject this module for exporting the Scheduler events data to an ICS file.
- **ICalendarImport** - Inject this module for importing the Scheduler events data from an ICS file.

Module injection

The required modules should be injected into the Scheduler using the `ej.schedule.Schedule.Inject` method of Scheduler within the `index.js` file as shown below. On doing so, only the injected module functionalities will be loaded and can be worked with Scheduler.

[myapp/index.js]

```
`ts
```

```
ej.schedule.Schedule.Inject(ej.schedule.Day, ej.schedule.Week, ej.schedule.WorkWeek,
ej.schedule.Month, ej.schedule.Agenda, ej.schedule.MonthAgenda);
`
```

Note: This module injection is not necessary in JavaScript.

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Scheduler interactions in EJ2 JavaScript Schedule control

The following table describes the Scheduler actions and also illustrates how those actions are carried out through mouse and touch interactions on Scheduler.

Actions	Mouse interaction	Touch interaction
Single click or tap on cells	Single click on a cell to select a cell. Tapping on it again will open the new event editor window.	Single tapping on cells, will display a + icon on the cell. Tapping on it again will open the new event editor window.
Multiple cell selection	Single click on a cell and drag the selection to other cells to enable multiple cell selection. No multiple cell selection is allowed using touch gestures.	
Event selection	Single click on an event to select it. Tap holding on events, select an event and opens a small popup at the top holding the options to edit or delete. The popup also displays the selected event's subject.	
Multiple event selection and deletion	Pressing Ctrl key and altogether single clicking on multiple events one after the other will enable multiple event selection. Pressing Delete key after event selection will delete all the selected events. Tap hold an event to select it, which opens a small popup at the top holding the options to edit or delete. As a continuation of this action, keep on single tapping on other events, to enable multiple event selection. Also, the popup displayed at the top remains in opened state, showing the count of the number of selected events. Pressing Delete option from the popup will delete all the selected events.	
Date navigation	Clicking on the previous or next date navigation icons in the header bar allows you to navigate between dates. Swiping the scheduler view port to the left or right will allow you to navigate between the dates on touch devices. You can prevent the swiping action by disabling allowSwiping property. NOTE: Swiping does not work when horizontal scroller present in the Scheduler. You can also make use of the previous and next navigation icons at the header bar to navigate.	
View navigation	Click on an event and try moving it over the Scheduler to enable drag and drop action. The view options are available within the popup options at the top right extreme end of the header bar and you can choose the view from it.	
Drag and drop	Click on an event and try moving it over the Scheduler to enable drag and drop action. Tap hold the event and try moving it over the Scheduler to enable drag and drop action.	
Event resizing	Hover the mouse across the extremities or edges of the Scheduler events and when the mouse pointer changes into resize handler, now click and start resizing an event to the desired time range. Touch the event extremities and start resizing the events directly.	

| Tooltip | Hover the mouse pointer over the events or resource header and the tooltip will be displayed. | Tap holding the events will open the tooltip on events. |

| Open editor window | Double click on cells or events to open the editor window. | Double click on cells or events to open the editor window. Single tap on cells, which displays a **+** icon on the cell. Now, tap on it again to open the new event editor window. To open the editor on events, single tap on it and then click on the edit icon to open the editor window in **Edit** mode. |

| Open quick info popup | Single clicking on a cell will open a quick popup prompting for new event creation. Single clicking on an event will open a popup displaying event information along with the option to edit and delete it. | No quick info popup is available while single tapping on cells. Single tapping on events, opens the popup showing event information. |

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Appointments in EJ2 JavaScript Schedule control

Appointments can be anything that are scheduled for a specific time period. It can be created on varied time range and each appointments are categorized based on this range. The Scheduler events can be categorized as,

- Normal events
- Spanned events
- All-day events
- Recurring events

Normal events

Represents an appointment that is created for any specific time interval within a day.

Creating a normal event

The following example depicts how to define a normal event on the Scheduler, with event data being loaded from simple JSON data.

INDEX.TS

```
import { Schedule, Day, Week, TimelineViews, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
  Subject: 'Paris',
  StartTime: new Date(2018, 1, 15, 10, 0),
  EndTime: new Date(2018, 1, 15, 12, 30)
}];
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
  eventSettings: {
    dataSource: data
  }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Spanned events

Represents an appointment that is created for more than 24 hours, and usually displayed on the all-day row. Also, represents another type of appointment that is created for more than one day but less than 24 hours, and usually displayed appropriately on both the days.

For example, if an appointment is created for two days say from November 25, 2018 – 11.00 PM to November 26, 2018 2.00 AM but less than 24 hours time interval, then the appointment is split into two partitions and will be displayed on both the days.

All-day events

Represents an appointment that is created for an entire day such as holiday events. It is usually displayed separately in an all-day row, a separate row for all-day appointments below the date header section. In Timeline views, the all-day appointments displays in the working space area, and no separate all-day row is present in that view.

To change normal appointment into all-day event, set `isAllDay` field to true.

Hide all-day row events

You can make use of the CSS customization to prevent the display of all-day row appointments on the Scheduler UI.

```
`ts
<style>
.e-schedule .e-date-header-wrap .e-schedule-table thead {
display: none;
}
</style>
`
```

You can also enable scroller for all-day row, please [refer](#) here to know more.

Expand all day appointments view on initial load

When you have larger number of appointments in all-day view, you can show all all-day events using `dataBound` event on at initial load. So, user don't have to click the toggle to expand all-day events.

INDEX.TS

```
import {Schedule, Day, Week, TimelineViews, Month, Agenda, DragAndDrop} from
'@syncfusion/ej2-schedule';
let data: object[] = [
  {
    EndTime: new Date(2022, 4, 4, 0, 0),
    Id: '1',
    IsAllDay: true,
    StartTime: new Date(2022, 4, 2, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Install Bullnose Brick/ Coping
| Jones | 3521',
  },
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: '2',
    IsAllDay: true,
    StartTime: new Date(2022, 3, 29, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Plumbing Checklist | Jaimungal
| 3671 :: Pool',
  },
  {
    EndTime: new Date(2022, 4, 7, 0, 0),
    Id: '3',
```

```

    IsAllDay: true,
    StartTime: new Date(2022, 4, 2, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Waterline Tile | Jaimungal |
3671 :: Pool',
  },
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: '4',
    IsAllDay: true,
    StartTime: new Date(2022, 3, 28, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Underground Plumbing |
Jaimungal | 3671 :: Pool',
  },
  {
    EndTime: new Date(2022, 4, 4, 0, 0),
    Id: '5',
    IsAllDay: true,
    StartTime: new Date(2022, 4, 3, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Backfill/ Compaction |
Jaimungal | 3671 :: Pool',
  },
  {
    EndTime: new Date(2022, 4, 7, 0, 0),
    Id: '6',
    IsAllDay: true,
    StartTime: new Date(2022, 4, 4, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Install Bullnose Brick/ Coping
| Jaimungal | 3671 :: Pool',
  },
  {
    EndTime: new Date(2022, 4, 1, 0, 0),
    Id: '7',
    IsAllDay: true,
    StartTime: new Date(2022, 3, 30, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Steel/ Checklist | VP Highland
Model | 3719 :: Pool',
  },
  {
    Description:
      'Let Yves know I did not see skimmer southern gunite did shell',
    EndTime: new Date(2022, 4, 4, 0, 0),
    Id: '8',
    IsAllDay: true,
    StartTime: new Date(2022, 4, 3, 0, 0),
    Subject:
      '<i class="fas fa-truck-pickup"></i> | Shotcrete Shell | VP Highland
Model | 3719 :: Pool',
  },
  {
    EndTime: new Date(2022, 3, 30, 0, 0),
    Id: '9',
    IsAllDay: true,

```



```

        StartTime: new Date(2022, 3, 29, 0, 0),
        Subject:
            '<i class="fas fa-truck-pickup"></i> | Tile Selections/ Pavers/ Finish
| VP Highland Model | 3719 :: Pool',
    },
    {
        EndTime: new Date(2022, 3, 30, 0, 0),
        Id: '10',
        IsAllDay: true,
        StartTime: new Date(2022, 3, 26, 0, 0),
        Subject:
            '<i class="fas fa-truck-pickup"></i> | Layout/ Form Rebar Shell | VP
Highland Model | 3719 :: Pool',
    },
];
var initialLoad = true;
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda, DragAndDrop);
let scheduleObj: Schedule = new Schedule({
    dataBound() {
        if (initialLoad) {
            this.element.querySelector('.e-all-day-appointment-section').click();
            initialLoad = false;
        }
    },
    height: '550px',
    selectedDate: new Date(2022, 3, 25),
    views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
    eventSettings: {
        dataSource: data,
    },
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the rendering of the spanned events

By default, Scheduler will renders the spanned events (appointment with more than 24 hours duration) in the all-day row by setting `AllDayRow` will the default type renders to the `spannedEventPlacement` option within the `eventSettings` property. Now we can customize rendering of the that events inside the work cells itself by modifying the `spannedEventPlacement` option as `TimeSlot`. In this following example, shows how to render the spanned appointments inside the work cells as follows.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
let data: object [] = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 17, 12, 30),
    IsAllDay: false
}, {
    Id: 2,
    Subject: 'London',
    StartTime: new Date(2018, 1, 16, 12, 0),
    EndTime: new Date(2018, 1, 18, 13, 0),
    IsAllDay: false
}];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: {
        dataSource: data,
        spannedEventPlacement: 'TimeSlot'
    }
});

```

```

    }
  });
  scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Recurring events

Represents an appointment that is created for a certain time interval and occurring repeatedly on a daily, weekly, monthly or yearly basis at the same time interval based on the provided recurrence rule. Usually, the recurring events are indicated by a repeat marker added at the bottom-right position.

Creating a recurring event

The following example depicts how to create a recurring event on Scheduler with the specific recurrence rule. In the following example, an event is made to repeat on daily mode and ends after 5 occurrences.

INDEX.TS

```
import { Schedule, Day, Week, TimelineMonth, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=5'
}];
Schedule.Inject(Day, Week, TimelineMonth, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'TimelineMonth', 'Month', 'Agenda'],
    eventSettings: {
        dataSource: data
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding exceptions

A few instance of the recurrence series can be excluded on specific dates, by adding those exceptional dates to the `recurrenceException` field. These date values should be given in the ISO date time format with no hyphens(-) separating the date elements.

For example, 22nd February 2018 can be represented as 20180222. Also, the time part being represented in UTC format needs to add "Z" after the time portion with no space. "07:30:00 UTC" is therefore represented as "073000Z".

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
    Id: 1,
    Subject: 'Paris',
    StartTime: new Date(2018, 0, 28, 10, 0),
    EndTime: new Date(2018, 0, 28, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=8',
    RecurrenceException: '20180129T043000Z,20180131T043000Z,20180202T043000Z'
}];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 0, 28),
    eventSettings: {
        dataSource: data
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Editing an occurrence from a series

To dynamically edit a particular occurrence from an event series and display it on the initial load of Scheduler, the edited occurrence needs to be added as a new event to the `dataSource` collection, with an additional `recurrenceID` field defined to it. The `recurrenceID` field of edited occurrence usually maps the ID value of the parent event.

In this example, a recurring instance that displays on the date 30th Jan 2018 is edited with different timings. Therefore, this particular date is excluded from the parent recurring event that repeats from 28th January 2018 to 4th February 2018. This can be done by adding the `recurrenceException` field with the excluded date value on the parent event. Also, the edited occurrence event which is created as a new event should carry the `recurrenceID` field pointing to the parent event's `Id` value.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
    Id: 1,
    Subject: 'Scrum Meeting',
    StartTime: new Date(2018, 0, 28, 10, 0),
    EndTime: new Date(2018, 0, 28, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=8',
    RecurrenceException: '20180130T043000Z'
},
{
    Id: 2,
    Subject: 'Scrum Meeting',
    StartTime: new Date(2018, 0, 30, 9, 0),
    EndTime: new Date(2018, 0, 30, 10, 30),
    Description: "Meeting time changed based on team activities.",
    RecurrenceID: 1
}
];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 0, 28),
    eventSettings: {
        dataSource: data
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Edit only the current and following events

To edit only the current and following events enable the property `editFollowingEvents` within `eventSettings` property. The edited occurrence needs to be added as a new event to the `dataSource` collection, with an additional `followingID` field defined to it. The `followingID` field of edited occurrence usually maps the ID value of the immediate parent event.

In this example, a recurring instance that displays on the date 30th Jan 2018 and its following dates are edited with different subject. Therefore, this particular date and its following dates are excluded from the parent recurring event that repeats from 28th January 2018 to 4th February 2018. This can be done by updating the `recurrenceRule` field with the until date value on the parent event. Also, the edited events which is created as a new event should carry the `followingID` field pointing to the immediate parent event's `Id` value.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
let data: object[] = [{
    Id: 1,
    Subject: 'Scrum Meeting',
    StartTime: new Date(2018, 0, 28, 10, 0),
    EndTime: new Date(2018, 0, 28, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;UNTIL=20180129T043000Z;',
},
{
    Id: 2,
    Subject: 'Scrum Meeting - Following Edited',
    StartTime: new Date(2018, 0, 30, 10, 0),
    EndTime: new Date(2018, 0, 30, 12, 30),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;UNTIL=20180204T043000Z;',
}

```



```

    FollowingID: 1
  }];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 0, 28),
  eventSettings: {
    dataSource: data,
    editFollowingEvents: true
  }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Recurrence options and rules

Events can be repeated on a daily, weekly, monthly or yearly basis based on the recurrence rule which accepts the string value. The following details should be assigned to the `recurrenceRule` property to generate the recurring instances.

- Repeat type - daily/weekly/monthly/yearly.
- How many times it needs to be repeated?
- The interval duration.
- The time period to render the appointment, etc.

There are four repeat types available namely,

- **Daily** - Creates the recurring instances on daily basis.
- **Weekly** - Creates the recurring instances on weekly basis for the selected days.
- **Monthly** - Creates the recurring instances on monthly basis for the selected months and other provided recurrence criteria.
- **Yearly** - Creates the recurring instances on yearly basis.

Recurrence properties

The properties based on which the recurrence appointments are created with its respective time period are depicted in the following table. Also, the valid rule string can be referred from [iCalendar](#) specifications.

Refer [iCalendar](#) specifications for valid recurrence rule string.

Property	Purpose	Example
FREQ	Maintains the repeat type (Daily, Weekly, Monthly, Yearly) value of the appointment.	FREQ=DAILY;INTERVAL=1
INTERVAL	Maintains the interval value of the appointments. When you create the daily appointment at an interval of 2, the appointments are rendered on the days Monday, Wednesday and Friday (Creates an appointment on all days by leaving the interval of one day gap).	FREQ=DAILY;INTERVAL=2
COUNT	It holds the appointment's count value. When the COUNT value is 10, then 10 instances of appointments are created in the recurrence series.	FREQ=DAILY;INTERVAL=1;COUNT=10
UNTIL	This property holds the end date value (in ISO format) denoting when the recurrence actually ends.	FREQ=DAILY;INTERVAL=1;UNTIL=20180530T041343Z;
BYDAY	It holds the day value(s), representing on which the appointments actually renders. Create the weekly appointment, and select the day(s) from the day options (Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday). When Monday is selected, the first two letters of the selected day "MO" is saved in the BYDAY property. When multiple days are selected, the values are separated by commas.	FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE;COUNT=10

| BYMONTHDAY | This property is used to store the date value of the Month, while creating the Month recurrence appointment. When you create a Monthly recurrence appointment for every 3rd day of the month, then BYMONTHDAY holds the value 3 and creates the appointment on 3rd day of every month. | FREQ=MONTHLY;BYMONTHDAY=3;INTERVAL=1;COUNT=10|

| BYMONTH | This property is used to store the index value of the selected Month while creating the yearly appointments. When you create the yearly appointment on June month, the index value of June month 6 will get stored in the BYMONTH field. The appointment is created on every 6th month of a year. | FREQ=YEARLY;BYMONTHDAY=16;BYMONTH=6;INTERVAL=1;COUNT=10|

| BYSETPOS | This property is used to store the index value of the week. When you create the monthly appointment in second week of a month, the index value of the second week (2) is stored in BYSETPOS. | FREQ=MONTHLY;BYDAY=MO;BYSETPOS=2;COUNT=10|

The default recurrence related validation has been included for recurrence appointments similar to the one available in Outlook. The validation usually occurs during the recurrence appointment creation, editing, drag and drop or resizing of the recurrence appointments and also if any single occurrence changes.

Daily Frequency

| Description | Example |

|-----|-----|

| Daily recurring event that never ends | FREQ=DAILY;INTERVAL=1 |

| Daily recurring event that ends after 5 occurrences | FREQ=DAILY;INTERVAL=1;COUNT=5 |

| Daily recurring event that ends exactly on 12/12/2018 |
FREQ=DAILY;INTERVAL=1;UNTIL=20181212T041343Z |

| Daily event that recurs on alternative days and repeats for 10 occurrences |
FREQ=DAILY;INTERVAL=2;COUNT=10 |

Weekly Frequency

| Description | Example |

|-----|-----|

| Weekly recurring event that repeats on every Monday, Wednesday and Friday and never ends |
FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE,FR |

| Repeats every week Thursday and ends after 10 occurrences | FREQ=WEEKLY;INTERVAL=1;BYDAY=TH;
COUNT=10 |

| Repeats every week Monday and ends on 12/12/2018 | FREQ=WEEKLY;INTERVAL=1;BYDAY=MO;
UNTIL=20181212T041343Z |

| Repeats on Monday, Wednesday and Friday of alternative weeks and ends after 10 occurrences |
FREQ=WEEKLY;INTERVAL=2;BYDAY=MO,WE,FR;COUNT=10 |

Monthly Frequency

| Description | Example |

|-----|-----|

| Monthly recurring event that repeats on every 15th day of a month and never ends | FREQ=MONTHLY;
BYMONTHDAY=15;INTERVAL=1 |

| Monthly recurring event that repeats on every 16th day of a month and ends after 10 occurrences |
FREQ=MONTHLY;BYMONTHDAY=16;INTERVAL=1;COUNT=10 |

| Repeats every 17th day of a month and ends on 12/12/2018 | FREQ=MONTHLY;BYMONTHDAY=17;
INTERVAL=1;UNTIL=20181212T041343Z |

| Repeats every 2nd Friday of a month and never ends | FREQ=MONTHLY;BYDAY=FR;BYSETPOS=2;
INTERVAL=1 |

| Repeats every 4th Wednesday of a month and ends after 10 occurrences |
FREQ=MONTHLY;BYDAY=WE; BYSETPOS=4;INTERVAL=1;COUNT=10 |

| Repeats every 4th Friday of a month and ends on 12/12/2018 |
FREQ=MONTHLY;BYDAY=FR;BYSETPOS=4; INTERVAL=1;UNTIL=20181212T041343Z; |

Yearly Frequency

| Description | Example |

|-----|-----|

| Yearly event that repeats on every 15th day of December month and never ends | FREQ=YEARLY;
BYMONTHDAY=15;BYMONTH=12;INTERVAL=1 |

| Event that repeats on every 10th day of December month and ends after 10 occurrences |
FREQ=YEARLY; BYMONTHDAY=10;BYMONTH=12;INTERVAL=1;COUNT=10 |

| Repeats on every 12th day of December month and ends on 12/12/2025 |
FREQ=YEARLY;BYMONTHDAY=12; BYMONTH=12;INTERVAL=1;UNTIL=20251212T041343Z |

| Repeats on every 3rd Friday of December month and never ends |
FREQ=YEARLY;BYDAY=FR;BYMONTH=12; BYSETPOS=3;INTERVAL=1 |

| Repeats on every 3rd Tuesday of December month and ends after 10 occurrences | FREQ=YEARLY;
BYDAY=TU;BYMONTH=12;BYSETPOS=3;INTERVAL=1;COUNT=10 |

| Repeats on every 4th Wednesday of December month and ends on 12/12/2028 |
FREQ=YEARLY;BYDAY=WE; BYMONTH=12;BYSETPOS=4;INTERVAL=1;UNTIL=20281212T041343Z |

Recurrence Validation

The built-in validation support has been added by default for recurring appointments during its creation, edit, drag and drop or resize action. The following are the possible validation alerts that displays on Scheduler while creating or editing the recurring events.

| Validation messages | Description |

|-----|-----|

| The recurrence pattern is not valid. | This alert will raise, when the selected recurrence rule value is not a valid one. For example, when you try to select the end date value (using **Until** option) for a recurring event, which occurs before the start date, an alert will popup out saying that the chosen pattern is invalid. |

| The changes made to specific instances of this series will be cancelled and those events will match the series again. | This alert will raise, when you try to edit the whole series, whose occurrence might have been already edited. For example, If there are five occurrences and one of the occurrence is already edited. Now, when you try to edit the entire series, you will get this validation alert. |

| The duration of the event must be shorter than how frequently it occurs. Shorten the duration, or change the recurrence pattern in the recurrence event editor. | This validation will occur, if the event duration is longer than the selected frequency. For example, if you create a recurring appointment with two days duration in **Daily** frequency with no intervals set to it, you may get this alert. |

| Some months have fewer than the selected date. For these months, the occurrence will fall on the last date of the month. | When you try to create a recurring appointment on 31st of every month, where few months won't have 31 days and in this scenario, you will get this alert. |

| Two occurrences of the same event cannot occur on the same day. | This validation will occur, when you try to edit or move any single occurrence to some other date, where another occurrence of the same event is already present. |

Event fields

The Scheduler dataSource usually holds the event instances, where each of the instance includes a collection of appropriate [fields](#). It is mandatory to map these fields with the equivalent fields of database, when remote data is bound to it. When the local JSON data is bound, then the field names defined within the instances needs to be mapped with the scheduler event fields correctly.

To create an event on Scheduler, it is enough to define the **startTime** and **endTime**. Also **id** field becomes mandatory to process CRUD actions on appropriate events.

Built-in fields

The built-in fields available on Scheduler event object are as follows.

Field name	Description
id	The id field needs to be defined as mandatory and this field usually assigns a unique ID value to each of the events.
subject	The subject field is optional, and usually assigns the summary text to each of the events.
startTime	The startTime field defines the start time of an event and it is mandatory to provide it for any of the valid event objects.
endTime	The endTime field defines the end time of an event and it is mandatory to provide the end time for any of the valid event objects.
starttimezone	It maps the starttimezone field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the startTime field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler.
endtimezone	It maps the endtimezone field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the endTime field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler.
location	It maps the location field from the dataSource and the location text value will be displayed over the events.
description	It maps the description field from the dataSource and denotes the event description which is optional.

| isAllDay | The `isAllDay` field is mapped from the `dataSource` and is used to denote whether an event is created for an entire day or for specific time alone. Usually, an event with `isAllDay` field set to true will be considered as an all-day event. |

| recurrenceID | It maps the `recurrenceID` field from `dataSource` and usually holds the ID value of the parent recurrence event. This field is applicable only for the edited occurrence events. |

| recurrenceRule | It maps the `recurrenceRule` field from `dataSource` and holds the recurrence rule value in a string format. Also, it uniquely identifies whether the event belongs to a recurring type or normal ones. |

| recurrenceException | It maps the `recurrenceException` field from `dataSource` and is used to hold the collection of exception dates, on which the recurring occurrences needs to be excluded. The `recurrenceException` should be specified in UTC format. |

| isReadOnly | It maps the `isReadOnly` field from `dataSource`. It is mainly used to make specific appointments as readonly when set to true. |

| isBlock | It maps the `isBlock` field from `dataSource`. It is used to block the particular time ranges in the Scheduler and prevents the event creation on those time slots. |

Binding different field names

When the fields of event instances has the default mapping name, it is not mandatory to map them manually. If a Scheduler's `dataSource` holds the events collection with different field names, then it is necessary to map them with its equivalent field name within the `eventSettings` property.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
let data: object [] = [{
  TravelId: 2,
  TravelSummary: 'Paris',
  DepartureTime: new Date(2018, 1, 15, 10, 0),
  ArrivalTime: new Date(2018, 1, 15, 12, 30),
  FullDay: false,
  Source: 'London',
  Comments: 'Summer vacation planned for outstation.',
  Origin: 'Asia/Yekaterinburg',
  Destination: 'Asia/Yekaterinburg'
}];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: {
    dataSource: data,
    fields: {
      id: 'TravelId',
      subject: { name: 'TravelSummary' },
      isAllDay: { name: 'FullDay' },
      location: { name: 'Source' },
      description: { name: 'Comments' },
      startTime: { name: 'DepartureTime' },
      endTime: { name: 'ArrivalTime' },
      starttimezone: { name: 'Origin' },
```

```

        endTimezone: { name: 'Destination' }
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The mapper field `id` is of string type and has no additional validation options, whereas all other fields are of `Object` type and has additional options.

Event field settings

Each field of the Scheduler events are provided with additional settings such as options to set default value, to map with appropriate data source fields, to validate every event fields and to provide label values for those fields in the event window.

Options	Description
default	Accepts the default value to the applicable fields (Subject, Location and Description), when no values are provided to them from dataSource.
name	Accepts the field name to be mapped from the dataSource fields.
title	Accepts the label values to be displayed for the fields of event editor.
validation	Defines the validation rules to be applied on the event fields within the event editor.

In following example, the Subject field in event editor will display its appropriate label as **Summary**. When no subject value is provided while saving an event, then the appointment will be saved with the default subject value as **Add Summary**.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
    TravelId: 2,
    TravelSummary: 'Paris',
    DepartureTime: new Date(2018, 1, 15, 10, 0),
    ArrivalTime: new Date(2018, 1, 15, 12, 30),
    Source: 'London',
    Comments: 'Summer vacation planned for outstation.'
}];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: {
        dataSource: data,
        fields: {
            id: 'TravelId',
            subject: { name: 'TravelSummary', title: 'Summary', default:
 'Add Summary' },
            location: { name: 'Source', default: 'USA' },
            description: { name: 'Comments' },
            startTime: { name: 'DepartureTime' },
            endTime: { name: 'ArrivalTime' }
        }
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding Custom fields

Apart from the default Scheduler fields, the user can include 'n' number of custom fields for appointments. The following code example shows how to include two custom fields namely **Status** and **Priority** within event collection. It is not necessary to bind the custom fields within the `eventSettings`. However, those additional fields can be accessed easily, for internal processing as well as from application end.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
let data: object [] = [{
    Id: 2,

```

```

        Subject: 'Meeting',
        StartTime: new Date(2018, 1, 15, 10, 0),
        EndTime: new Date(2018, 1, 15, 12, 30),
        IsAllDay: false,
        Status: 'Completed',
        Priority: 'High'
    }];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: {
        dataSource: data,
        fields: {
            id: 'Id',
            subject: { name: 'Subject' },
            isAllDay: { name: 'IsAllDay' },
            startTime: { name: 'StartTime' },
            endTime: { name: 'EndTime' }
        }
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the order of the overlapping events

By default, the scheduler will render the overlapping events based on the start and end time. Now we can customize the order of the overlapping events based on the custom fields by using the `sortComparer` property grouped under the `eventSettings` property. The following code example shows how to sort the appointments based on the custom field as follows.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
    Id: 1,
    Subject: 'Rank 1',
    StartTime: new Date(2017, 9, 29, 10, 0),
    EndTime: new Date(2017, 9, 29, 11, 30),
    IsAllDay: false,
    RankId: '1'
}, {
    Id: 2,
    Subject: 'Rank 3',
    StartTime: new Date(2017, 9, 29, 10, 30),
    EndTime: new Date(2017, 9, 29, 12, 30),
    IsAllDay: false,
    RankId: '3'
}, {
    Id: 3,
    Subject: 'Rank 6',
    StartTime: new Date(2017, 9, 29, 7, 0),
    EndTime: new Date(2017, 9, 29, 14, 30),
    IsAllDay: false,
    RankId: '6'
}, {
    Id: 4,
    Subject: 'Rank 9',
    StartTime: new Date(2017, 9, 29, 11, 0),
    EndTime: new Date(2017, 9, 29, 15, 30),
    IsAllDay: false,
    RankId: '9'
}
];
let comparerFun: SortComparerFunction = (args: Record<string, any>) =>

```

```

args.sort((event1: Record<string, any>, event2: Record<string, any>) =>
    event1.RankId.localeCompare(event2.RankId, undefined, { numeric: true })
);
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda );
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2017, 9, 29),
    eventSettings: {
        dataSource: data,
        sortComparer: comparerFun
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop appointments

Appointments can be rescheduled to any time by dragging and dropping them onto the desired location. To work with drag and drop functionality, it is necessary to inject the module **DragAndDrop** and make sure that **allowDragAndDrop** is set to true on Scheduler. In mobile mode, you can drag and drop the events by tap holding an event and dropping them on to the desired location.

By default, drag and drop action is applicable on all Scheduler views, except Agenda, Month-Agenda and Year view.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, TimelineViews, TimelineMonth, Month, Agenda,
DragAndDrop } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, TimelineViews, TimelineMonth, Month, Agenda,
DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['TimelineDay', 'Day', 'Week', 'TimelineMonth', 'Month',
'Agenda'],
    eventSettings: { dataSource: data }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop multiple appointments

We can drag and drop multiple appointments by enabling the `allowMultiDrag` property. We can select multiple appointments by holding the CTRL key. Once the events are selected, we can leave the CTRL key and start dragging the event.

We can also drag multiple events from one resource to another resource. In this case, if all the selected events are in the different resources, then all the events should be moved to the single resource that is related to the target event.

Note: Multiple events drag and drop is not supported on mobile devices.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    allowMultiDrag: true,
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable the drag action

By default, you can drag and drop the events within any of the applicable scheduler views, and to disable it, set `false` to the `allowDragAndDrop` property.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from '../datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({

```

```

        height: '550px',
        allowDragAndDrop: false,
        selectedDate: new Date(2018, 1, 15),
        eventSettings: { dataSource: data },
    });
    scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```


Preventing drag and drop on specific targets

It is possible to prevent the drag action on particular target, by passing the target to be excluded in the `excludeSelectors` option within `dragStart` event. In this example, we have prevented the drag action on all-day row.

INDEX.TS

```
import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    dragStart: (args: DragEventArgs) => {
        args.excludeSelectors = 'e-header-cells,e-header-day,e-header-
date,e-all-day-cells';
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable scrolling on drag action

By default, while dragging an appointment to the edges, either top/bottom in the vertical Scheduler or left/right in the timeline Scheduler, scrolling action takes place automatically. To prevent this scrolling, set `false` to the `scroll` value within the `dragStart` event arguments.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    dragStart: (args: DragEventArgs) => {
        args.scroll = { enable: false };
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Controlling scroll speed while dragging an event

The speed of the scrolling action while dragging an appointment to the Scheduler edges, can be controlled within the `dragStart` event by setting the desired value to the `scrollBy` and `timeDelay` option whereas its default value is 30 minutes and 100ms.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    dragStart: (args: DragEventArgs) => {
        args.scroll = { enable: true, scrollBy: 5, timeDelay: 200 };
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Auto navigation of date ranges on dragging an event

When an event is dragged either to the left or right extreme edges of the Scheduler and kept hold for few seconds without dropping, the auto navigation of date ranges will be enabled allowing the Scheduler to navigate from current date range to back and forth respectively. This action is set to **false** by default and to enable it, you need to set **navigation** to true within the **dragStart** event.

By default, the navigation delay is set to 2000ms. The navigation delay decides how long the user needs to drag and hold the appointments at the extremities. You can also set your own delay value for letting the users to navigate based on it, using the **timeDelay** within the **dragStart** event.

INDEX.TS

```
import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    dragStart: (args: DragEventArgs) => {
        args.navigation = { enable: true, timeDelay: 4000 }
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Setting drag time interval

By default, while dragging an appointment, it moves at an interval of 30 minutes. To change the dragging time interval, pass the appropriate values to the `interval` option within the `dragStart` event.

INDEX.TS

```
import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    dragStart: (args: DragEventArgs) => {
        args.interval = 10; //drag interval time is changed to 10 minutes
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Schedule Typescript Component</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop items from external source

It is possible to drag and drop the unplanned items from any of the external source into the scheduler, by manually saving those dropped item as a new appointment data through `addEvent` method of Scheduler.

In this example, we have used the tree view control as an external source and the child nodes from the tree view component are dragged and dropped onto the Scheduler. Therefore, it is necessary to make use of the `nodeDragStop` event of the TreeView component, where we can form an event object and save it using the `addEvent` method.

INDEX.TS

```

import { Schedule, Month, Resize, DragAndDrop, CellClickEventArgs } from
'@syncfusion/ej2-schedule';
import { ActionEventArgs } from '@syncfusion/ej2-schedule';
import { eventData, waitingList } from './datasource.ts';
import { DragAndDropEventArgs, TreeView } from '@syncfusion/ej2-
navigations';
import { closest, remove, addClass } from '@syncfusion/ej2-base';
Schedule.Inject(Month, Resize, DragAndDrop);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    cssClass: 'schedule-drag-drop',
    views: [ 'Month' ],
    eventSettings: { dataSource: eventData },
    actionBegin: onActionBegin,
    drag: onItemDrag
});
scheduleObj.appendTo('#Schedule');

```

```

let treeObj: TreeView = new TreeView({
    fields: { dataSource: waitingList , id: 'Id', text: 'Name' },
    allowDragAndDrop: true,
    nodeDragStop: onTreeDragStop,
    nodeDragging: onItemDrag,
    cssClass: 'treeview-external-drag'
});
treeObj.appendTo('#Tree');
let isTreeItemDropped: boolean = false;
let draggedItemId: string = '';
function onItemDrag(event: any): void {
    if (scheduleObj.isAdaptive) {
        let classElement: HTMLElement =
scheduleObj.element.querySelector('.e-device-hover');
        if (classElement) {
            classElement.classList.remove('e-device-hover');
        }
        if (event.target.classList.contains('e-work-cells')) {
            addClass([event.target], 'e-device-hover');
        }
    }
    if (document.body.style.cursor === 'not-allowed') {
        document.body.style.cursor = '';
    }
    if (event.name === 'nodeDragging') {
        let dragElementIcon: NodeListOf<HTMLElement> =
document.querySelectorAll('.e-drag-item.treeview-external-drag
.e-icon-expandable');
        for (let i: number = 0; i < dragElementIcon.length; i++) {
            dragElementIcon[i].style.display = 'none';
        }
    }
}
function onActionBegin(event: ActionEventArgs): void {
    if (event.requestType === 'eventCreate' && isTreeItemDropped) {
        let treeViewdata: { [key: string]: Object }[] =
treeObj.fields.dataSource as { [key: string]: Object }[];
        const filteredPeople: { [key: string]: Object }[] =
            treeViewdata.filter((item: any) => item.Id !==
parseInt(draggedItemId, 10));
        treeObj.fields.dataSource = filteredPeople;
        let elements: NodeListOf<HTMLElement> =
document.querySelectorAll('.e-drag-item.treeview-external-drag');
        for (let i: number = 0; i < elements.length; i++) {
            remove(elements[i]);
        }
    }
}
function onTreeDragStop(event: DragAndDropEventArgs): void {
    let treeElement: Element = <Element>closest(event.target, '.e-
treeview');
    let classElement: HTMLElement = scheduleObj.element.querySelector('.e-
device-hover');
    if (classElement) {
        classElement.classList.remove('e-device-hover');
    }
    if (!treeElement) {

```



```

        event.cancel = true;
        let scheduleElement: Element = <Element>closest(event.target, '.e-
content-wrap') ||
            <Element>closest(event.target, '.e-all-day-row');
        if (scheduleElement) {
            let treeviewData: { [key: string]: Object }[] =
                treeObj.fields.dataSource as { [key: string]: Object }[];
            if (event.target.classList.contains('e-work-cells')) {
                const filteredData: { [key: string]: Object }[] =
                    treeviewData.filter((item: any) => item.Id ===
parseInt(event.draggedNodeData.id as string, 10));
                let cellData: CellClickEventArgs =
scheduleObj.getCellDetails(event.target);
                let eventData: { [key: string]: Object } = {
                    Subject: filteredData[0].Name,
                    StartTime: cellData.startTime,
                    EndTime: cellData.endTime,
                    IsAllDay: cellData.isAllDay,
                    Description: filteredData[0].Description
                };
                scheduleObj.addEvent(eventData);
                isTreeItemDropped = true;
                draggedItemId = event.draggedNodeData.id as string;
            }
        }
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>

    <div id="container" class="container col-lg-12 custom-tree">
        <div class="content-wrapper">
            <div class="schedule-container">
                <div class="title-container">
                    <h3 class="title-text">Scheduler</h3>
                </div>
                <div id="Schedule">
                </div>
            </div>
            <div class="treeview-container">
                <div class="title-container">
                    <h3 class="title-text">Waiting List</h3>
                </div>
                <div id="Tree"></div>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Opening the editor window on drag stop

There are scenarios where you want to open the editor filled with data on newly dropped location and may need to proceed to save it, only when **Save** button is clicked on the editor. On clicking the cancel button should revert these changes. This can be achieved using the **dragStop** event of Scheduler.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop,
DragEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    dragStop: (args: DragEventArgs) => {
        args.cancel = true; //cancels the drop action
        scheduleObj.openEditor(args.data, "Save"); //open the event window
        //with updated start and end time
    }
});

```

```
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Inline Appointment

In Scheduler, another easier way for adding or editing the appointment's subject alone can be achieved by using the inline Add/Edit support. It allows the user to add and edit the appointments inline. To get familiar with the inline Add mode, single click on any of the Scheduler cells or press enter key on the selected cells.

When the inline adding mode is ON, a text box will get created within the clicked Scheduler cells with a blinking cursor in it, requiring the user to enter the subject of an appointment. Once the subject is entered, the appointment will be saved on pressing the enter key.

To enable the inline edit mode, single click on any of the existing appointment's subject, so that the user can edit the subject of that appointment. The edited subject of that appointment will be updated on pressing the enter key.

The inline option can be enabled/disabled on the Scheduler by using the `allowInline` API, whereas its default value is set to false.

While using the `allowInline` the `showQuickInfo` will be turned off. The `quickPopup` will not show on clicking the work cell or clicking the appointment when the `allowInline` property is set to true.

In work cells, select multiple cells using keyboard, and then press enter key. The appointment wrapper will be created, and focus will be on the subject field. Also, consider the overlapping scenarios when creating an inline event.

Normal Event

While editing appointments, single-click the appointment subject, the `editable` option will be enabled in UI and the cursor will focus at the end of the text. Inline editing will be considered for all possible views.

Recurrence Event

While editing the occurrence from the recurrence series, it is only possible to edit a `single occurrence`, not an entire series.

INDEX.TS

```
import { Schedule, Day, Week, TimelineViews, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
  Subject: 'Paris',
  StartTime: new Date(2018, 1, 15, 10, 0),
  EndTime: new Date(2018, 1, 15, 12, 30)
},{
  Subject: 'Scrum Meeting',
  StartTime: new Date(2018, 1, 11, 9, 30),
  EndTime: new Date(2018, 1, 11, 11, 0)
},{
  Subject: 'Meeting',
  StartTime: new Date(2018, 1, 13, 10, 0),
  EndTime: new Date(2018, 1, 13, 10, 30)
},{
  Subject: 'Inline Editor Window',
  StartTime: new Date(2018, 1, 13, 11, 0),
  EndTime: new Date(2018, 1, 13, 12, 0)
},{
  Subject: 'Validated',
  StartTime: new Date(2018, 1, 17, 11, 0),
  EndTime: new Date(2018, 1, 17, 12, 30)
}];
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 11),
  views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
```

```

        allowInline: true,
        eventSettings: {
            dataSource: data
        }
    });
    scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Appointment Resizing

Another way of rescheduling an appointment can be done by resizing it through either of its handlers. To work with resizing functionality, it is necessary to inject the module `Resize` and make sure that `allowResizing` property is set to true.

INDEX.TS

```
import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, Resize } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Resize);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
```

```

        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable the resize action

By default, resizing of events is allowed on all Scheduler views except Agenda and Month-Agenda view. To disable this event resizing action, set false to the `allowResizing` property.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, Resize } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Resize);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    allowResizing: false,
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable scrolling on resize action

By default, while resizing an appointment, when its handler reaches the extreme edges of the Scheduler, scrolling action will take place along with event resizing. To prevent this scrolling action, set false to `scroll` value within the `resizeStart` event.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, Resize,
ResizeEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Resize);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    resizeStart: (args: ResizeEventArgs) => {
        args.scroll = { enable: false };
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">

```



```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Controlling scroll speed while resizing an event

The speed of the scrolling action while resizing an appointment to the Scheduler edges, can be controlled within the `resizeStart` event by setting the desired value to the `scrollBy` option.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, Resize,
ResizeEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Resize);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    resizeStart: (args: ResizeEventArgs) => {

```

```

        args.scroll = { enable: true, scrollBy: 15 };
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting resize time interval

By default, while resizing an appointment, it extends or shrinks at an interval of 30 minutes. To change this default resize interval, set appropriate values to `interval` option within the `resizeStart` event.

INDEX.TS

```
import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, Resize,
ResizeEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Resize);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data },
    resizeStart: (args: ResizeEventArgs) => {
        args.interval = 10; //resize interval time is changed to 10 minutes
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Appointment customization

The look and feel of the Scheduler events can be customized using any one of the following ways.

- [Using event templates](#)
- [Using eventRendered event](#)
- [Using custom CSS class](#)

Using template

Any kind of text, images and links can be added to customize the look of the events. The user can format and change the default appearance of the events by making use of the **template** option available within the **eventSettings** property. The following code example customizes the appointment's default color and time format.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { webinarData } from './datasource.ts';
import { Internationalization } from '@syncfusion/ej2-base';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let instance: Internationalization = new Internationalization();
(window as TemplateFunction).getTimeString = (value: Date) => {
  return instance.formatDate(value, { skeleton: 'hm' });
};
interface TemplateFunction extends Window {
  getTimeString?: Function;
}
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '500px',
  readonly: true,
  selectedDate: new Date(2018, 1, 15),
  eventSettings: {
    dataSource: webinarData,
    template: '#apptemplate'
  }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-schedule .e-vertical-view .e-content-wrap .e-appointment {
        border-radius: 8px;
    }
    .e-schedule .e-vertical-view .e-content-wrap .e-appointment .e-
appointment-details {
        padding: 0;
        height: 100%;
    }
    .e-schedule .template-wrap {
        height: 100%;
        white-space: normal;
    }
    .e-schedule .template-wrap .subject {
        font-weight: 600;
        font-size: 15px;
        padding: 4px 4px 4px;
        text-overflow: ellipsis;
        white-space: nowrap;
        overflow: hidden;
    }
    .e-schedule .template-wrap .time {
        height: 50px;
        font-size: 12px;
        padding: 4px 6px 4px;
        overflow: hidden;
    }
    .e-past-app {
        background-color: chocolate !important;
    }
    .custom-class.e-schedule .e-vertical-view .e-appointment {
        background: green;
    }
</style>
<script id="apptemplate" type="text/x-template">

```

```

<div class="template-wrap" style="background:${SecondaryColor}">
  <div class="subject"
style="background:${PrimaryColor}">${Subject}</div>
  <div class="time" style="background:${PrimaryColor}">Time:
${getTimeString(data.StartTime)} - ${getTimeString(data.EndTime)}</div>
</div>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

All the built-in fields that are mapped to the appropriate field properties within the `eventSettings`, as well as custom mapped fields from the Scheduler `dataSource` can be accessed within the template code.

Using `eventRendered` event

The `eventRendered` event triggers before the appointment renders on the Scheduler. Therefore, this client-side event can be utilized to customize the look of events based on any specific criteria, before rendering them on the scheduler.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda View,
EventRenderedArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  width: '100%',
  cssClass: 'custom-class',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: { dataSource: scheduleData },
  eventRendered: (args: EventRenderedArgs) => applyCategoryColor(args,
scheduleObj.currentView)
});
scheduleObj.appendTo('#Schedule');
function applyCategoryColor(args: EventRenderedArgs, currentView: View):
void {
  let categoryColor: string = args.data.CategoryColor as string;
  if (!args.element || !categoryColor) {
    return;
  }
}

```

```

    }
    if (scheduleObj.currentView === 'Agenda') {
        (args.element.firstChild as HTMLElement).style.borderLeftColor =
categoryColor;
    } else {
        args.element.style.backgroundColor = categoryColor;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using custom CSS class

The customization of events can also be achieved using `cssClass` property of the Scheduler. In the following example, the background of appointments has been changed using the `cssClass`.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    width: '100%',
    cssClass: 'custom-class',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

  <div id="container">
```



```

        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting minimum height

It is possible to set minimal height for appointments on Scheduler using `eventRendered` event, when its start and end time duration is less than the default duration of a single slot.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, ActionEventArgs } from
'@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, WorkWeek, Month);
let data: object[] = [{
    Id: 13,
    Subject: 'Myths of Andromeda Galaxy',
    StartTime: new Date(2018, 1, 16, 10, 30),
    EndTime: new Date(2018, 1, 16, 10, 40)
}, {
    Id: 14,
    Subject: 'Aliens vs Humans',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 10, 20)
}];
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    width: '100%',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: { dataSource: data },
    eventRendered: (args: EventRenderedArgs) => {
        let cellHeight: number = (scheduleObj.element.querySelector('.e-
work-cells') as HTMLElement).offsetHeight;
        let appHeight: number = (args.data.EndTime.getTime() -
args.data.StartTime.getTime()) / (60 * 1000) * (cellHeight *
scheduleObj.timeScale.slotCount) / scheduleObj.timeScale.interval;
        args.element.style.height = appHeight + 'px';
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Block Dates and Times

It is possible to block a set of dates or a particular time ranges on the Scheduler. To do so, define an appointment object within `eventSettings` along with the required time range to block and set the `isBlock` field to true. Usually, the event objects defined with `isBlock` field set to true will block the entire time cells lying within the appropriate time ranges specified through `startTime` and `endTime` fields.

INDEX.TS

```

import { Schedule, Day, Week, TimelineViews, Month, Agenda, Resize,
DragAndDrop } from '@syncfusion/ej2-schedule';
import { blockData } from './datasource.ts';
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda, Resize,
DragAndDrop);
let scheduleObj: Schedule = new Schedule({
    height: '550px',

```

```

selectedDate: new Date(2018, 1, 15),
views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
eventSettings: {
    dataSource: blockData
}
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Block events can also be defined to repeat on several days as shown in the following code example.

INDEX.TS

```
import { Schedule, Day, Week, TimelineViews, Month, Agenda, Resize,
DragAndDrop } from '@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda, Resize,
DragAndDrop);
let data: object [] = [{
    Id: 1,
    Subject: 'Explosion of Betelgeuse Star',
    StartTime: new Date(2018, 1, 15, 9, 30),
    EndTime: new Date(2018, 1, 15, 11, 0),
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=5',
    IsBlock: true
}, {
    Id: 2,
    Subject: 'Thule Air Crash Report',
    StartTime: new Date(2018, 1, 14, 12, 0),
    EndTime: new Date(2018, 1, 14, 14, 0)
}];
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
    eventSettings: {
        dataSource: data
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Readonly

An interaction with the appointments of Scheduler can be enabled/disabled using the **readonly** property. With this property enabled, you can simply navigate between the Scheduler dates, views and can be able to view the appointment details in the quick info window. Most importantly, the users are not allowed to perform any CRUD actions on Scheduler, when this property is set to true. By default, it is set as **false**.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    readonly: true,
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Make specific events readonly

There are scenarios where you need to restrict the CRUD action on specific appointments alone based on certain conditions. In the following example, the events that has occurred on the past hours from the current date of the Scheduler are made as read-only and the CRUD actions has been prevented only on those appointments. This can be achieved by setting `isReadOnly` field of read-only events to `true`.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { readOnlyData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    width: '100%',
    views: ['Day', 'Week', 'WorkWeek'],
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: readOnlyData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

DATASOURCE.TS

```

export let readOnlyData: Object[] = [
  {
    Id: 1,
    Subject: 'Explosion of Betelgeuse Star',
    StartTime: new Date(2018, 1, 11, 9, 30),
    EndTime: new Date(2018, 1, 11, 11, 0),
    IsReadOnly: true
  }, {
    Id: 2,

```

```

        Subject: 'Thule Air Crash Report',
        StartTime: new Date(2018, 1, 12, 12, 0),
        EndTime: new Date(2018, 1, 12, 14, 0),
        IsReadOnly: true
    }, {
        Id: 3,
        Subject: 'Blue Moon Eclipse',
        StartTime: new Date(2018, 1, 13, 9, 30),
        EndTime: new Date(2018, 1, 13, 11, 0),
        IsReadOnly: true
    }, {
        Id: 4,
        Subject: 'Meteor Showers in 2018',
        StartTime: new Date(2018, 1, 14, 13, 0),
        EndTime: new Date(2018, 1, 14, 14, 30),
        IsReadOnly: true
    }, {
        Id: 5,
        Subject: 'Milky Way as Melting pot',
        StartTime: new Date(2018, 1, 15, 12, 0),
        EndTime: new Date(2018, 1, 15, 14, 0),
        IsReadOnly: true
    }, {
        Id: 6,
        Subject: 'Mysteries of Bermuda Triangle',
        StartTime: new Date(2018, 1, 15, 9, 30),
        EndTime: new Date(2018, 1, 15, 11, 0),
        IsReadOnly: false
    }, {
        Id: 7,
        Subject: 'Glaciers and Snowflakes',
        StartTime: new Date(2018, 1, 16, 11, 0),
        EndTime: new Date(2018, 1, 16, 12, 30),
        IsReadOnly: false
    }, {
        Id: 8,
        Subject: 'Life on Mars',
        StartTime: new Date(2018, 1, 17, 9, 0),
        EndTime: new Date(2018, 1, 17, 10, 0),
        IsReadOnly: false
    }, {
        Id: 9,
        Subject: 'Alien Civilization',
        StartTime: new Date(2018, 1, 11, 12, 0),
        EndTime: new Date(2018, 1, 11, 14, 0),
        IsReadOnly: true
    }, {
        Id: 10,
        Subject: 'Wildlife Galleries',
        StartTime: new Date(2018, 1, 21, 11, 0),
        EndTime: new Date(2018, 1, 21, 13, 0),
        IsReadOnly: false
    }, {
        Id: 11,
        Subject: 'Best Photography 2018',
        StartTime: new Date(2018, 1, 22, 9, 30),
        EndTime: new Date(2018, 1, 22, 11, 0),

```



```

        IsReadOnly: false
    }, {
        Id: 12,
        Subject: 'Smarter Puppies',
        StartTime: new Date(2018, 1, 9, 10, 0),
        EndTime: new Date(2018, 1, 9, 11, 30),
        IsReadOnly: true
    }, {
        Id: 13,
        Subject: 'Myths of Andromeda Galaxy',
        StartTime: new Date(2018, 1, 7, 10, 30),
        EndTime: new Date(2018, 1, 7, 12, 30),
        IsReadOnly: true
    }, {
        Id: 14,
        Subject: 'Aliens vs Humans',
        StartTime: new Date(2018, 1, 5, 10, 0),
        EndTime: new Date(2018, 1, 5, 11, 30),
        IsReadOnly: true
    }, {
        Id: 15,
        Subject: 'Facts of Humming Birds',
        StartTime: new Date(2018, 1, 20, 9, 30),
        EndTime: new Date(2018, 1, 20, 11, 0),
        IsReadOnly: false
    }, {
        Id: 16,
        Subject: 'Sky Gazers',
        StartTime: new Date(2018, 1, 23, 11, 0),
        EndTime: new Date(2018, 1, 23, 13, 0),
        IsReadOnly: false
    }, {
        Id: 17,
        Subject: 'The Cycle of Seasons',
        StartTime: new Date(2018, 1, 12, 5, 30),
        EndTime: new Date(2018, 1, 12, 7, 30),
        IsReadOnly: true
    }, {
        Id: 18,
        Subject: 'Space Galaxies and Planets',
        StartTime: new Date(2018, 1, 12, 17, 0),
        EndTime: new Date(2018, 1, 12, 18, 30),
        IsReadOnly: true
    }, {
        Id: 19,
        Subject: 'Lifecycle of Bumblebee',
        StartTime: new Date(2018, 1, 15, 6, 0),
        EndTime: new Date(2018, 1, 15, 7, 30),
        IsReadOnly: false
    }, {
        Id: 20,
        Subject: 'Sky Gazers',
        StartTime: new Date(2018, 1, 15, 16, 0),
        EndTime: new Date(2018, 1, 15, 18, 0),
        IsReadOnly: false
    }
];

```

By default, the event editor is prevented to open on the read-only events when `isReadOnly` field is set to `true`.

Restricting event creation on specific time slots

You can restrict the users to create and update more than one appointment on specific time slots. Also, you can disable the CRUD action on those time slots if it is already occupied, which can be achieved using Scheduler's public method `isSlotAvailable`.

Note: The `isSlotAvailable` is centered around verifying appointments within the present view's date range. Yet, it does not encompass an evaluation of availability for recurrence occurrences that fall beyond this particular date range.

INDEX.TS

```
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { Schedule, Day, TimelineViews, WorkWeek, Month, ActionEventArgs }
from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, TimelineViews, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    width: '100%',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'TimelineWeek', 'WorkWeek', 'Month'],
    currentView: 'TimelineWeek',
    eventSettings: { dataSource: scheduleData },
    actionBegin: (args: ActionEventArgs) => {
        if ((args.requestType === 'eventCreate' || args.requestType ===
        'eventChange') && (<Object[]>args.data).length > 0 ||
        !isNullOrUndefined(args.data)) {
            let eventData: any = args.data as any;
            let eventField: EventFieldsMapping = scheduleObj.eventFields;
            let startDate: Date = (((<Object[]>args.data).length > 0) ?
            eventData[0][eventField.startTime] : eventData[eventField.startTime]) as
            Date;
            let endDate: Date = (((<Object[]>args.data).length > 0) ?
            eventData[0][eventField.endTime] : eventData[eventField.endTime]) as Date;
            args.cancel = !scheduleObj.isSlotAvailable(startDate, endDate);
        }
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
    base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Differentiate the past time events

To differentiate the appearance of the appointments based on specific criteria such as displaying the past hour appointments with different colors on Scheduler, `eventRendered` event can be used which triggers before the appointment renders on the Scheduler.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, ActionEventArgs } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let isReadOnly: Function = (data: { [key: string]: Object }): boolean => {
    return (data.EndTime < scheduleObj.selectedDate);
};
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    width: '100%',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData },

```

```

    eventRendered: (args: EventRenderedArgs) => {
        if (isReadOnly(args.data)) {
            args.element.classList.add('e-past-app');
        }
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Appointments occupying entire cell

The Scheduler allows the event to occupies the full height of the cell without its header part by setting `true` for `enableMaxHeight` Property.

We can show more indicator if more than one appointment is available in a same cell by setting `true` to `enableIndicator` property whereas its default value is false.

INDEX.TS

```
import { Schedule, TimelineViews, TimelineMonth } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '500px',
    currentView: 'TimelineMonth',
    views: ['TimelineWeek', 'TimelineMonth'],
    selectedDate: new Date(2018, 1, 15),
    eventSettings: {
        dataSource: scheduleData,
        enableMaxHeight: true,
        enableIndicator: false
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to limit maximum number of events to display

In the Scheduler, the default behavior is to display concurrent events based on cell height, with each new event represented as

+n more characters. However, you may want to improve the quality of the presentation by limiting the number of concurrent events. This can be accomplished by using the [maxEventsPerRow](#) property, which is defaulted to the [views](#) property.

The [maxEventsPerRow](#) property is specific to the month, timeline month, and timeline year views, allowing you to view events visually in these rows. Below is a code example that demonstrates how to use this constraint and the events displayed in a cell have been created:

INDEX.TS

```

import { Schedule, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '380px',
    selectedDate: new Date(2023, 11, 15),
    views: [{ option: 'Month', maxEventsPerRow: 3 }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The property [maxEventsPerRow](#) will be applicable only when [rowAutoHeight](#) feature is disabled in the Scheduler.

Display tooltip for appointments

The tooltip shows the Scheduler appointment's information in a formatted style by making use of the tooltip related options.

Show or hide built-in tooltip

The tooltip can be displayed for appointments by setting `true` to the `enableTooltip` option within the `eventSettings` property.

INDEX.TS

```

import { Schedule, TimelineViews, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({

```

```

width: '100%',
height: '500px',
views: ['TimelineDay', 'Week', 'WorkWeek', 'Month', 'Agenda'],
selectedDate: new Date(2018, 1, 15),
eventSettings: {
    dataSource: scheduleData,
    enableTooltip: true
}
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```



```
</body></html>
```

Customizing event tooltip using template

After enabling the default tooltip, it is possible to customize the display of needed event information on tooltip by making use of the `tooltipTemplate` option within the `eventSettings`.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { eventsData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let template: string = '<div class="tooltip-wrap">' +
    '<div class="content-area"><div class="name">${Subject}</div>' +
    '${if(City !== null && City !== undefined)}<div
class="city">${City}</div>${if}' +
    '<div class="time">From : ${StartTime.toLocaleString()} </div>' +
    '<div class="time">To : ${EndTime.toLocaleString()}
</div></div></div>';
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: {
        dataSource: eventsData,
        enableTooltip: true,
        tooltipTemplate: template
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-schedule-event-tooltip .tooltip-wrap .name {
        font-weight: 500;
        font-size: 14px;
    }

    .e-schedule-event-tooltip .tooltip-wrap {
        display: flex;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

All the field names that are mapped from the Scheduler dataSource to the appropriate field properties such as subject, description, location, startTime and endTime within the `eventSettings` can be accessed within the template.

Appointment filtering

The appointments can be filtered by passing the predicate value to `query` option in `eventSettings`. The following code example shows how to filter and render the selected appointments alone in the Scheduler.

INDEX.TS

```

import { CheckBox } from '@syncfusion/ej2-buttons';
import { Query, Predicate } from '@syncfusion/ej2-data';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, EventRenderedArgs }
from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),

```

```

eventSettings: { dataSource: scheduleData },
eventRendered: (args: EventRenderedArgs) => {
  switch (args.data.EventType) {
    case 'Requested':
      (args.element as HTMLElement).style.backgroundColor = '#F57F17';
      break;
    case 'Confirmed':
      (args.element as HTMLElement).style.backgroundColor = '#7fa900';
      break;
    case 'New':
      (args.element as HTMLElement).style.backgroundColor = '#8e24aa';
      break;
  }
};
scheduleObj.appendTo('#Schedule');
function onChange(): void {
  let predicate: Predicate;
  let checkBoxes: CheckBox[] = [confirm, request, fresh];
  checkBoxes.forEach((checkBoxObj: CheckBox) => {
    if (checkBoxObj.checked) {
      if (predicate) {
        predicate = predicate.or('EventType', 'equal',
checkBoxObj.label);
      } else {
        predicate = new Predicate('EventType', 'equal',
checkBoxObj.label);
      }
    }
  });
  scheduleObj.eventSettings.query = new Query().where(predicate);
}
let confirm: CheckBox = new CheckBox({ label: 'Confirmed', checked: true,
change: onChange }, '#confirmed');
let request: CheckBox = new CheckBox({ label: 'Requested', checked: true,
change: onChange }, '#requested');
let fresh: CheckBox = new CheckBox({ label: 'New', checked: true, change:
onChange }, '#new');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table id="property" title="Filter events">
            <tbody>
                <tr>
                    <td>
                        <input id="confirmed" type="checkbox">
                    </td>
                    <td>
                        <input id="requested" type="checkbox">
                    </td>
                    <td>
                        <input id="new" type="checkbox">
                    </td>
                </tr>
            </tbody>
        </table>
        <div id="Schedule"></div>
    </div>
    <style>
        #property td {
            padding: 0 5px;
        }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Appointment selection

Appointment selection can be done either through mouse or keyboard actions. The selected events in UI will have a box shadow effect around to differentiate it from other appointments.

| Action | Description |

|-----|-----|

| Mouse click or Single tap on appointments | Selects single appointment. |

| Ctrl + [Mouse click] or [Single tap] on appointments | Selects multiple appointments. |

Deleting multiple appointments

With the options available to select multiple appointments, it is also possible to delete the multiple selected appointments simply by pressing the **delete** key. In case of deleting multiple selected occurrences of an event series, only those occurrences will be deleted and not the entire series.

Retrieve event details from the UI of an event

It is possible to access the information about the event fields of an appointment element displayed on the Scheduler UI. This can be achieved by passing an appointment element as argument to the public method **getEventDetails**.

In the following example, the subject of the appointment clicked has been displayed.

INDEX.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, EventClickArgs } from '@syncfusion/ej2-schedule';
import { eventData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: eventData },
    eventClick: onEventClick
});
scheduleObj.appendTo('#Schedule');
let btn: Button = new Button();
btn.appendTo('#clear');
document.getElementById('clear').onclick = () => {
    document.getElementById('EventLog').innerHTML = '';
};
function onEventClick(args: EventClickArgs): void {
    let event: Object = scheduleObj.getEventDetails(args.element);
    appendElement(event.Subject + '<hr>');
}
function appendElement(html: string): void {
    let span: HTMLElement = document.createElement('span');
    span.innerHTML = html;
    let log: HTMLElement = document.getElementById('EventLog');
    log.insertBefore(span, log.firstChild);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="col-lg-12">
        <div class="content-wrapper">
            <div class="col-lg-9 control-section">
                <div id="Schedule"></div>
            </div>
            <div class="col-lg-3 property-section">
                <table id="property" title="Event Trace">
                    <tbody>
                        <tr>
                            <td>
                                <div class="eventarea" style="height:
245px;overflow: auto">
                                    <span class="EventLog" id="EventLog"
style="word-break: normal;"></span>
                                </div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div class="evtbtn" style="padding-bottom:
10px">
                                    <input id="clear" type="button"
value="Clear">
                                </div>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get the current view appointments

To retrieve the appointments present in the current view of the Scheduler, you can make use of the `getCurrentViewEvents` public method. In the following example, the count of current view appointment collection rendered has been traced in `dataBound` event.

INDEX.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from '@syncfusion/ej2-schedule';
import { eventData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: eventData },
    dataBound: onDataBound
});
scheduleObj.appendTo('#Schedule');
let btn: Button = new Button();
btn.appendTo('#clear');
document.getElementById('clear').onclick = () => {
    document.getElementById('EventLog').innerHTML = '';
};
function onDataBound(): void {
    let event: Object[] = scheduleObj.getCurrentViewEvents();
    if (event.length > 0) {
        appendElement('Events present on current view <b>' + event.length
+ '<b><hr>');
    } else {
        appendElement('No Events available in this view.<hr>');
    }
}
function appendElement(html: string): void {
    let span: HTMLElement = document.createElement('span');
    span.innerHTML = html;
    let log: HTMLElement = document.getElementById('EventLog');
    log.insertBefore(span, log.firstChild);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="col-lg-12">
        <div class="content-wrapper">
            <div class="col-lg-9 control-section">
                <div id="Schedule"></div>
            </div>
            <div class="col-lg-3 property-section">
                <table id="property" title="Event Trace">
                    <tbody>
                        <tr>
                            <td>
                                <div class="eventarea" style="height:
245px;overflow: auto">
                                    <span class="EventLog" id="EventLog"
style="word-break: normal;"></span>
                                </div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div class="evtbtn" style="padding-bottom:
10px">
                                    <input id="clear" type="button"
value="Clear">
                                </div>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>

```



```

        </tbody>
      </table>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Get the entire appointment collections

The entire collection of appointments rendered on the Scheduler can be accessed using the `getEvents` public method. In the following example, the count of entire appointment collection rendered on the Scheduler has been traced in `dataBound` event.

INDEX.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { eventData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: { dataSource: eventData },
  dataBound: onDataBound
});
scheduleObj.appendTo('#Schedule');
let btn: Button = new Button();
btn.appendTo('#clear');
document.getElementById('clear').onclick = () => {
  document.getElementById('EventLog').innerHTML = '';
};
function onDataBound(): void {
  let event: Object[] = scheduleObj.getEvents();
  appendElement('Events present on scheduler <b>' + event.length
+ '<b><hr>');
}
function appendElement(html: string): void {
  let span: HTMLElement = document.createElement('span');
  span.innerHTML = html;
  let log: HTMLElement = document.getElementById('EventLog');
  log.insertBefore(span, log.firstChild);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="col-lg-12">
        <div class="content-wrapper">
            <div class="col-lg-9 control-section">
                <div id="Schedule"></div>
            </div>
            <div class="col-lg-3 property-section">
                <table id="property" title="Event Trace">
                    <tbody>
                        <tr>
                            <td>
                                <div class="eventarea" style="height:
245px;overflow: auto">
                                    <span class="EventLog" id="EventLog"
style="word-break: normal;"></span>
                                </div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div class="evtbtn" style="padding-bottom:
10px">
                                    <input id="clear" type="button"
value="Clear">
                                </div>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>

```

```

        </tbody>
      </table>
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Refresh appointments

If your requirement is to simply refresh the appointments instead of refreshing the entire Scheduler elements from your application end, make use of the `refreshEvents` public method.

```
`ts
```

```
scheduleObj.refreshEvents();
```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Data binding in EJ2 JavaScript Schedule control

The Scheduler uses `DataManager`, which supports both RESTful data service binding and JavaScript object array binding. The `dataSource` property of Scheduler can be assigned either with the instance of `DataManager` or JavaScript object array collection, as it supports the following two kind of data binding methods:

- Local data
- Remote data

Binding local data

To bind local JSON data to the Scheduler, you can simply assign a JavaScript object array to the `dataSource` option of the scheduler within the `eventSettings` property. The JSON object `dataSource` can also be provided as an instance of `DataManager` and assigned to the Scheduler `dataSource` property.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleData: object[] = [{
  Id: 1,
  Subject: 'Explosion of Betelgeuse Star',
  StartTime: new Date(2018, 1, 15, 9, 30),
  EndTime: new Date(2018, 1, 15, 11, 0)
}, {
  Id: 2,

```

```

    Subject: 'Thule Air Crash Report',
    StartTime: new Date(2018, 1, 12, 12, 0),
    EndTime: new Date(2018, 1, 12, 14, 0)
  }, {
    Id: 3,
    Subject: 'Blue Moon Eclipse',
    StartTime: new Date(2018, 1, 13, 9, 30),
    EndTime: new Date(2018, 1, 13, 11, 0)
  }, {
    Id: 4,
    Subject: 'Meteor Showers in 2018',
    StartTime: new Date(2018, 1, 14, 13, 0),
    EndTime: new Date(2018, 1, 14, 14, 30)
  }
];
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

        <div id="container">
            <div id="Schedule"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, **DataManager** uses **JsonAdaptor** for binding local data.

You can also bind different field names to the default event fields as well as include additional custom fields to the event object collection which can be referred [here](#).

Binding remote data

Any kind of remote data services can be bound to the Scheduler. To do so, create an instance of **DataManager** and provide the service URL to the **url** option of **DataManager** and then assign it to the **dataSource** property within **eventSettings**.

Using ODataV4Adaptor

ODataV4 is a standardized protocol for creating and consuming data. Refer to the following code example to retrieve the data from ODataV4 service using the **DataManager**. To connect with ODataV4 service end points, it is necessary to make use of **ODataV4Adaptor** within **DataManager**.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let dataManager: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(1996, 6, 9),
    readonly: true,
    eventSettings: {
        dataSource: dataManager,
        fields: {
            id: 'Id',
            subject: { name: 'ShipName' },
            location: { name: 'ShipCountry' },
            description: { name: 'ShipAddress' },
            startTime: { name: 'OrderDate' },
            endTime: { name: 'RequiredDate' },
            recurrenceRule: { name: 'ShipRegion' }
        }
    }
});

```

```
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Filter events using the in-built query

To enable server-side filtering operations based on predetermined conditions, the [includeFiltersInQuery](#) API can be set to true, this allows the filter query to be constructed using the start date, end date, and recurrence rule which in turn enables the request to be filtered accordingly.

This method greatly improves the component's performance by reducing the data that needs to be transferred to the client side. As a result, the component's efficiency and responsiveness are significantly enhanced, resulting in a better user experience. However, it is important to consider the possibility of longer query strings, which may cause issues with the maximum URL length or server limitations on query string length.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let dataManager: DataManager = new DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
  adaptor: new ODataV4Adaptor,
  crossDomain: true
});
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(1996, 6, 9),
  currentView: 'Month',
  readonly: true,
  eventSettings: {
    query: new Query(),
    includeFiltersInQuery: true, dataSource: dataManager,
    fields: {
      id: 'Id',
      subject: { name: 'ShipName' },
      location: { name: 'ShipCountry' },
      description: { name: 'ShipAddress' },
      startTime: { name: 'OrderDate' },
      endTime: { name: 'RequiredDate' },
      recurrenceRule: { name: 'ShipRegion' }
    }
  }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following image represents how the parameters are passed using ODataV4 filter.

General

Request URL: https://services.odata.org/V4/Northwind/Northwind.svc/Orders/?\$filter=(((((OrderDate%20ge%201996-06-29T18:30:00.000Z)%20and%20(RequiredDate%20ge%201996-06-29T18:30:00.000Z))%20and%20(OrderDate%20lt%201996-08-03T18:30:00.000Z))%20or%20((OrderDate%20le%201996-06-29T18:30:00.000Z)%20and%20(RequiredDate%20gt%201996-06-29T18:30:00.000Z)))%20or%20((ShipRegion%20ne%20null)%20and%20(ShipRegion%20ne%20%27%27)))

Request Method: GET

Status Code: 200 OK

Remote Address: 137.117.17.70:443

Referrer Policy: strict-origin-when-cross-origin

Using custom adaptor

It is possible to create your own custom adaptor by extending the built-in available adaptors. The following example demonstrates the custom adaptor usage and how to add a custom field **EventID** for the appointments by overriding the built-in response processing using the **processResponse** method of the **ODataV4Adaptor**.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
class CustomAdaptor extends ODataV4Adaptor {
  processResponse(): Object {
    let i: number = 0;
    // calling base class processResponse function
    let original: Object[] = super.processResponse.apply(this,
arguments);
    // adding employee id
    original.forEach((item: Object) => item['EventID'] = ++i);
    return original;
  }
}
let dataManager: DataManager = new DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
  adaptor: new CustomAdaptor
});
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(1996, 6, 9),
  readonly: true,
  eventSettings: {
    dataSource: dataManager,
    fields: {
      id: 'Id',
      subject: { name: 'ShipName' },
      location: { name: 'ShipCountry' },
      description: { name: 'ShipAddress' },
      startTime: { name: 'OrderDate' },
      endTime: { name: 'RequiredDate' },
      recurrenceRule: { name: 'ShipRegion' }
    }
  }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Loading data via AJAX post

You can bind the event data through external ajax request and assign it to the `dataSource` property of Scheduler. In the following code example, we have retrieved the data from server with the help of ajax request and assigned the resultant data to the `dataSource` property of Scheduler within the `onSuccess` event of Ajax.

[src/app/app.ts]

```

`ts
import { Ajax } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from '@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let dataManager: object = [];
let ajax = new Ajax('Home/GetData', 'GET', false);

```

```

ajax.onSuccess = function (value) {
  dataManager = value;
};
ajax.send();
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2017, 5, 11),
  eventSettings: { dataSource: dataManager }
});
scheduleObj.appendTo('#Schedule');

```

Definition for the controller method `GetData` can be referred [here](#).

Passing additional parameters to the server

To send an additional custom parameter to the server-side post, you need to make use of the `addParams` method of `Query`. Now, assign this `Query` object with additional parameters to the `query` property of Scheduler.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let dataManager: DataManager = new DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/',
  adaptor: new ODataV4Adaptor
});
let dataQuery: Query = new Query().addParams('readOnly', 'true');
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(1996, 6, 9),
  readOnly: true,
  eventSettings: { dataSource: dataManager, query: dataQuery,
    fields: {
      id: 'Id',
      subject: { name: 'ShipName' },
      location: { name: 'ShipCountry' },
      description: { name: 'ShipAddress' },
      startTime: { name: 'OrderDate' },
      endTime: { name: 'RequiredDate' },
      recurrenceRule: { name: 'ShipRegion' }
    }
  }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The parameters added using the [query](#) property will be sent along with the data request sent to the server on every scheduler actions.

Handling failure actions

During the time of Scheduler interacting with server, there are chances that some server-side exceptions may occur. You can acquire those error messages or exception details in client-side using the [actionFailure](#) event of Scheduler.

The argument passed to the [actionFailure](#) event contains the error details returned from the server.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { DataManager } from '@syncfusion/ej2-data';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let dataManager: DataManager = new DataManager({
  url: 'http://some.com/invalidUrl'
});
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2017, 5, 11),
  eventSettings: { dataSource: dataManager },
  actionFailure: () => {
    let span: HTMLElement = document.createElement('span');
    scheduleObj.element.parentNode.insertBefore(span,
scheduleObj.element);
    span.style.color = '#FF0000'
    span.innerHTML = 'Server exception: 404 Not found';
  }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

        <div id="container">
            <div id="Schedule"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The [actionFailure](#) event will be triggered not only on server returning errors, but also when there is an exception while processing any of the Scheduler CRUD actions.

Scheduler CRUD actions

The CRUD (Create, Read, Update and Delete) actions can be performed easily on Scheduler appointments using the various adaptors available within the `DataManager`. Most preferably, we will be using `UrlAdaptor` for performing CRUD actions on scheduler appointments.

`ts

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from '@syncfusion/ej2-schedule';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';

Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);

let dataManager: DataManager = new DataManager({
    url: 'Home/GetData', // 'controller/actions'
    crudUrl: 'Home/UpdateData',
    adaptor: new UrlAdaptor
});

let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2017, 5, 5),
    eventSettings: { dataSource: dataManager }
});

scheduleObj.appendTo('#Schedule');
`

```

The server-side controller code to handle the CRUD operations are as follows.

`c#

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```
using System.Web;
using System.Web.Mvc;
using ScheduleSample.Models;
namespace ScheduleSample.Controllers
{
    public class HomeController : Controller
    {
        ScheduleDataDataContext db = new ScheduleDataDataContext();
        public ActionResult Index()
        {
            return View();
        }

        public JsonResult LoadData() // Here we get the Start and End Date and based on that can filter the data
        and return to Scheduler
        {
            var data = db.ScheduleEventDatas.ToList();
            return Json(data, JsonRequestBehavior.AllowGet);
        }

        [HttpPost]
        public JsonResult UpdateData(EditParams param)
        {
            if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
            will execute while inserting the appointments
            {
                var value = (param.action == "insert") ? param.value : param.added[0];
                int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
                1;
                DateTime startTime = Convert.ToDateTime(value.StartTime);
                DateTime endTime = Convert.ToDateTime(value.EndTime);
                ScheduleEventData appointment = new ScheduleEventData()
                {
                    Id = intMax + 1,
                    StartTime = startTime.ToLocalTime(),
                    EndTime = endTime.ToLocalTime(),
                }
            }
        }
    }
}
```

```
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
```



```
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
var data = db.ScheduleEventDatas.ToList();
return Json(data, JsonRequestBehavior.AllowGet);
}

public class EditParams
{
public string key { get; set; }
public string action { get; set; }
public List<ScheduleEventData> added { get; set; }
public List<ScheduleEventData> changed { get; set; }
public List<ScheduleEventData> deleted { get; set; }
public ScheduleEventData value { get; set; }
}
}
}
```

Configuring Scheduler with Google API service

We have assigned our custom created Google Calendar url to the DataManager and assigned the same to the Scheduler `dataSource`. Since the events data retrieved from the Google Calendar will be in its own object format, therefore it needs to be resolved manually within the Scheduler's `dataBinding` event. Within this event, the event fields needs to be mapped properly and then assigned to the result.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
const CALENDAR_ID: string = 'en.usa%23holiday@group.v.calendar.google.com';
const PUBLIC_KEY: string = 'AIzaSyBgbX_tgmVanBP4yafDPPXxWr70sjbKAXM';
let dataManager: DataManager = new DataManager({
  url: 'https://www.googleapis.com/calendar/v3/calendars/' + CALENDAR_ID +
    '/events?key=' + PUBLIC_KEY,
  adaptor: new WebApiAdaptor,
  crossDomain: true
});
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  eventSettings: { dataSource: dataManager },
  readonly: true,
  currentView: 'Month',
  timezone: 'UTC',
  dataBinding: (e: { [key: string]: Object }) => {
    let items: { [key: string]: Object }[] = (e.result as { [key:
string]: Object }).items as { [key: string]: Object }[];
    let scheduleData: Object[] = [];
    if (items.length > 0) {
      for (let i: number = 0; i < items.length; i++) {
        let event: { [key: string]: Object } = items[i];
        let when: string = (event.start as { [key: string]: Object
}).dateTime as string;
        let start: string = (event.start as { [key: string]: Object
}).dateTime as string;
        let end: string = (event.end as { [key: string]: Object
}).dateTime as string;
        if (!when) {
          when = (event.start as { [key: string]: Object }).date
as string;
          start = (event.start as { [key: string]: Object }).date
as string;
          end = (event.end as { [key: string]: Object }).date as
string;
        }
        scheduleData.push({
          Id: event.id,
          Subject: event.summary,
          StartTime: new Date(start),
          EndTime: new Date(end),
          IsAllDay: !(event.start as { [key: string]: Object
}).dateTime
```

```

        });
    }
    }
    e.result = scheduleData;
}
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

[Salesforce Integration](#)

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Crud actions in EJ2 JavaScript Schedule control

Events, a.k.a. Appointments, play an important role in Scheduler with which the users mostly interact. You can easily manipulate (add/edit/delete) the desired appointments as and when required either using the editor window or through the drag and resize action.

Add

Any kind of appointments such as normal, all-day, spanned or recurring events can be easily added on Scheduler using any one of the following ways.

- [Creation using editor window](#)
- [Creation using addEvent method](#)

Creation using editor window

The default editor window opens when you double click on the Scheduler cells. It provides you with event related options such as Subject, Location, Start and End time, All-day, Timezone, Description and other recurrence options. With these available fields, you can choose to provide detailed information to the events. Once the fields are filled with proper values, enter the **Save** button to add an event.

In case, if you want to simply provide the Subject alone for appointments, just single click on the required cells which will open the quick popup expecting you to enter subject alone and save it. You can also select multiple cells and press **Enter** key to open the quick popup for selected time range and save the appointment for that time range.

In case, if you need to add some other additional fields to the editor window, then you can opt for [custom editor window](#) which allows you to include fields as per your application needs. If you need to add just one or two [additional fields to the existing default editor window](#), you can do so by defining it manually and then appending it to the editor window.

Creation using addEvent method

The appointments can be created dynamically by using **addEvent** method. Either you can add a single or a collection of appointment objects using **addEvent** method. The following code example let you know how to use the **addEvent** method to create multiple appointments simultaneously.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleData: Object[] = [{
    Id: 1,
    Subject: 'Testing',
    StartTime: new Date(2018, 1, 11, 9, 0),
    EndTime: new Date(2018, 1, 11, 10, 0),
    IsAllDay: false
}, {
```

```

        Id: 2,
        Subject: 'Vacation',
        StartTime: new Date(2018, 1, 13, 9, 0),
        EndTime: new Date(2018, 1, 13, 10, 0),
        IsAllDay: false
    }];
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: {
        dataSource: scheduleData
    }
});
scheduleObj.appendTo('#Schedule');
let add: Button = new Button();
add.appendTo('#add');
add.element.onclick = (): void => {
let Data: Object[] = [{
    Id: 3,
    Subject: 'Conference',
    StartTime: new Date(2018, 1, 12, 9, 0),
    EndTime: new Date(2018, 1, 12, 10, 0),
    IsAllDay: true
}, {
    Id: 4,
    Subject: 'Meeting',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 11, 30),
    IsAllDay: false
}];
scheduleObj.addEvent(Data);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="add">Add</button>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Inserting events into database at server-side

While adding the normal or recurring events to the Scheduler, **insert** action takes place and the following code example describes how to add a new event into database at server side.

```
`ts
```

```

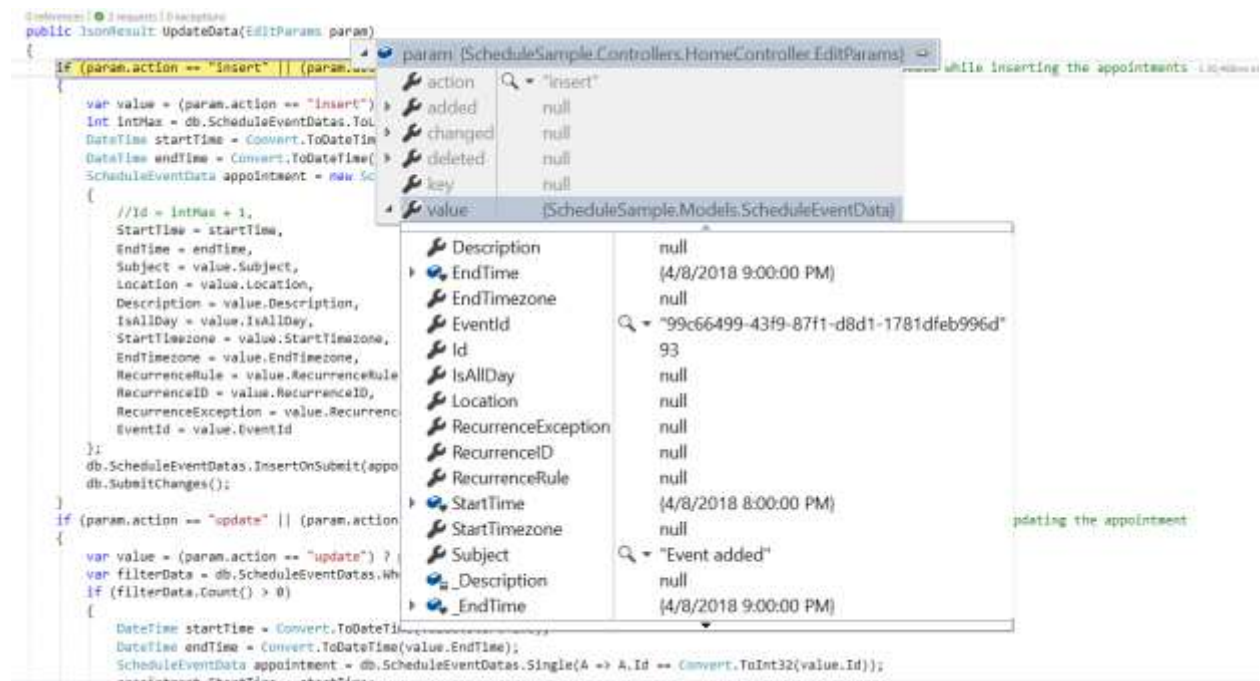
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
will execute while inserting the appointments
{
    var value = (param.action == "insert") ? param.value : param.added[0];
    int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
    1;
    DateTime startTime = Convert.ToDateTime(value.StartTime);
    DateTime endTime = Convert.ToDateTime(value.EndTime);
    ScheduleEventData appointment = new ScheduleEventData()
    {
        Id = intMax + 1,
        StartTime = startTime.ToLocalTime(),
        EndTime = endTime.ToLocalTime(),
        Subject = value.Subject,

```

```

IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}

```



Restricting add action based on specific criteria

In the following example, the specific fields of Scheduler editor window such as Subject and Location are made to undergo validation such that if it is left as blank, then the default **Required** validation message will be displayed, while clicking on a save button.

Additionally, the regex condition has been added to the Location field, so that if any special characters are typed into it, then the custom validation message will be displayed.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({

```

```

width: '100%',
height: '550px',
selectedDate: new Date(2018, 1, 15),
eventSettings: {
  dataSource: scheduleData,
  fields: {
    subject: { name: 'Subject', validation: { required: true } },
    location: {
      name: 'Location', validation: {
        required: true,
        regex: ["^[a-zA-Z0-9- ]*$", 'Special character(s) not
allowed in this field']
      }
    }
  }
}
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">

```



```

        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can also dynamically prevent the creation of appointments on Scheduler. For example, say if you want to decline the creation of appointments on weekend days, you can check for its appropriate condition within the `actionBegin` event.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda, ActionEventArgs }
from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData },
    actionBegin: (args: ActionEventArgs) => {
        let weekEnds: number[] = [0, 6];
        if (args.requestType == 'eventCreate' &&
            weekEnds.indexOf((args.data[0].StartTime).getDay()) >= 0) {
            args.cancel = true;
        }
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Edit

The same way the appointments such as normal, all-day, spanned or recurring events are created, it can be easily edited using any of the following ways.

- [Update using editor window](#)
- [Update using saveEvent method](#)

Update using editor window

You can open the default editor window filled with appointment details by double clicking on the required events. It gets pre-filled with event options such as Subject, Location, Start and End time, All-day, timezone, description and other recurrence options, from which you can edit the desired field values and, then enter the **Save** button to update it.

You can also single click on appointments, which opens the quick info popup with edit and delete options. Clicking on the **edit** option will open the default editor filled with event details and **delete** option will prompt for delete confirmation.

Update using saveEvent method

The appointments can be edited and updated manually using the **saveEvent** method. The following code examples shows how to edit the normal and recurring events.

Normal event - Here, an event with ID **3** is edited and its subject is changed with a new text. When the modified data object is passed onto the **saveEvent** method, the changes gets reflected onto the original

event. The **Id** field is mandatory in this edit process, where the modified event object should hold the valid **Id** value that exists in the Scheduler data source.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleData: Object[] = [{
    Id: 3,
    Subject: 'Testing',
    StartTime: new Date(2018, 1, 11, 9, 0),
    EndTime: new Date(2018, 1, 11, 10, 0),
    IsAllDay: false
}, {
    Id: 4,
    Subject: 'Vacation',
    StartTime: new Date(2018, 1, 13, 9, 0),
    EndTime: new Date(2018, 1, 13, 10, 0),
    IsAllDay: false
}];
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: {
        dataSource: scheduleData
    }
});
scheduleObj.appendTo('#Schedule');
let edit: Button = new Button();
edit.appendTo('#edit');
edit.element.onclick = (): void => {
    let Data: Object = {
        Id: 3,
        Subject: 'Testing-edited',
        StartTime: new Date(2018, 1, 11, 10, 0),
        EndTime: new Date(2018, 1, 11, 11, 0),
        IsAllDay: false
    };
    scheduleObj.saveEvent(Data);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="edit">Edit</button>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Recurring event - The following code example shows how to edit a single occurrence of a recurring event. In this case, the modified data should hold an additional field namely **RecurrenceID** mapping to its parent recurring event's Id value. Also, this modified occurrence will be considered as a new event in the Scheduler dataSource, where it is linked with its parent event through the **RecurrenceID** field value. The **saveEvent** method takes 2 arguments, first one accepting the modified event data object and second argument accepting either of the 2 text values - **EditOccurrence** or **EditSeries**.

When the second argument is passed as **EditOccurrence**, which means that the passed event data is a single modified occurrence - whereas if the second argument is passed as **EditSeries**, it means that the modified data needs to be edited as a whole series and therefore no new event object will be maintained in the Scheduler dataSource.

In case of modifying the single occurrence, it is also necessary to update the **RecurrenceException** field of parent event altogether with the occurrence editing. To know more about how to set **RecurrenceException** values, refer the [recurring events](#) topic.

INDEX.TS

```

import { DataManager, Query, Predicate } from '@syncfusion/ej2-data';
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleData: Object[] = [{
  Id: 3,
  Subject: 'Testing',
  StartTime: new Date(2018, 1, 11, 9, 0),
  EndTime: new Date(2018, 1, 11, 10, 0),
  IsAllDay: false,
  RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3'
}, {
  Id: 4,
  Subject: 'Vacation',
  StartTime: new Date(2018, 1, 12, 11, 0),
  EndTime: new Date(2018, 1, 12, 12, 0),
  IsAllDay: false,
  RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
}];
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'WorkWeek', 'Month'],
  eventSettings: {
    dataSource: scheduleData
  }
});
scheduleObj.appendTo('#Schedule');
let edit: Button = new Button();
edit.appendTo('#edit');
edit.element.onclick = (): void => {
  let data: Object = new
DataManager(scheduleObj.getCurrentViewEvents()).executeLocal(new
Query().where('RecurrenceID', 'equal', 3));
  data[0].Subject = 'Edited';
  scheduleObj.saveEvent(data[0], 'EditOccurrence');
  edit.element.setAttribute('disabled', 'true');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="edit">Edit</button>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Updating events in database at server-side

While editing the normal events in the Scheduler, **update** action takes place and the following code example describes how to update event into database at server side.

```

`ts

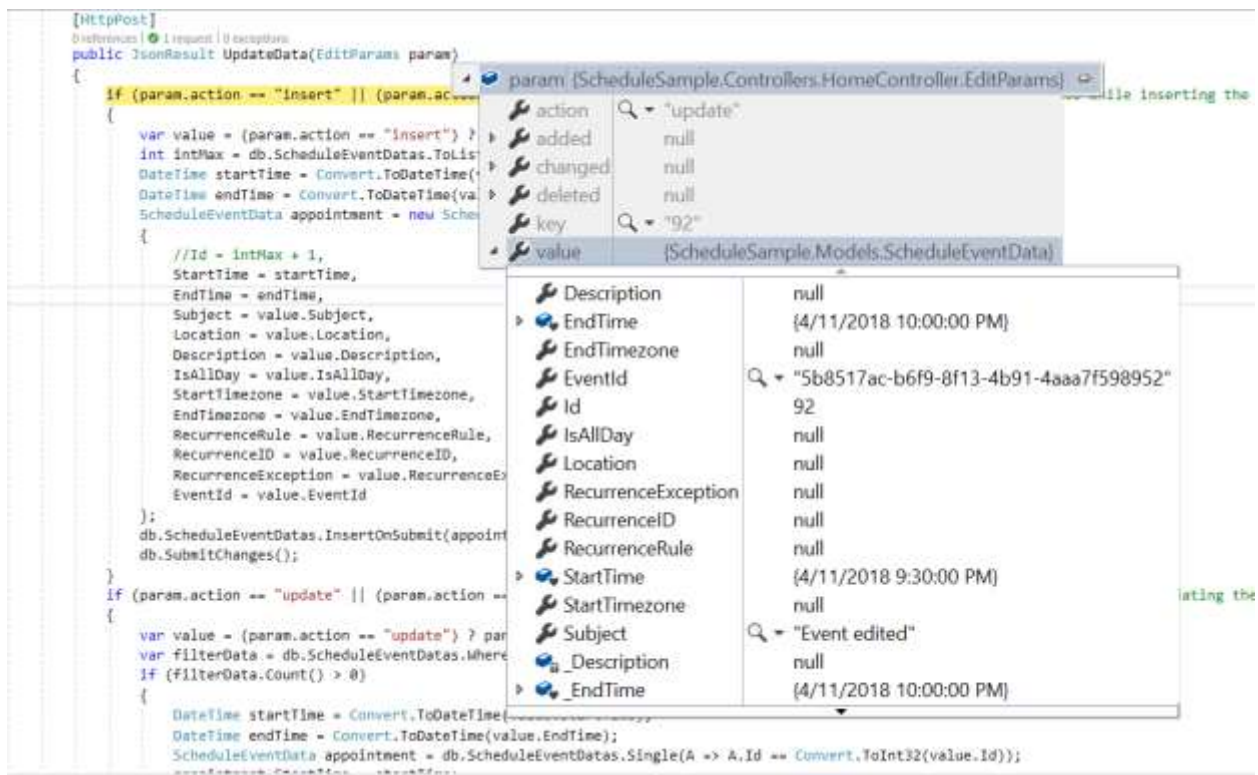
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
    var value = (param.action == "update") ? param.value : param.changed[0];
    var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
    if (filterData.Count() > 0)
    {
        DateTime startTime = Convert.ToDateTime(value.StartTime);
        DateTime endTime = Convert.ToDateTime(value.EndTime);
        ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));

```

```

appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}

```



How to edit a single occurrence or entire series and update it in database at server-side

The recurring appointments can be edited in either of the following two ways.

- Single occurrence
- Entire series

Editing single occurrence - When you double click on a recurring event, a popup prompts you to choose either to edit the single event or entire series. From this, if you choose to select **EDIT EVENT** option, a single occurrence of the recurring appointment alone will be edited. The following process takes place while editing a single occurrence,

- A new event will be created from the parent event data and added to the Scheduler dataSource, with all its default field values overwritten with the newly modified data and additionally, the **recurrenceID** field will be added to it, that holds the **id** value of the parent recurring event. Also, a new **Id** will be generated for this event in the dataSource.
- The parent recurring event needs to be updated with appropriate **recurrenceException** field to hold the edited occurrence appointment's date collection.

Therefore, when a single occurrence is edited from a recurring event, the batch action takes place by allowing both the **Add** and **Edit** action requests to take place together.

In case, if you edit an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get updated. In this case, **update** action alone takes place on the edited occurrence object on the database.

```
`ts
```

```
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
will execute while inserting the appointments
{
var value = (param.action == "insert") ? param.value : param.added[0];
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
Id = intMax + 1,
StartTime = startTime.ToLocalTime(),
EndTime = endTime.ToLocalTime(),
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
RecurrenceException = value.RecurrenceException
```



```

};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
`ts

```

Editing entire series - When you select an option **EDIT SERIES** from the popup that opens on double clicking the recurring event, the whole recurring series will be updated with the newly provided value. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences will be removed from the dataSource and simply the single parent data will be updated.

This action of editing entire series also leads to the batch process, as both the **Delete** and **Edit** action takes place together.

`ts

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment

```
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
```

if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of code will execute while removing the appointment

```
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
```

```

foreach (var apps in param.deleted)
{
    ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
    if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
,

```

To know more about handling recurrence exceptions, refer the [Adding exceptions](#) topic.

How to edit from the current and following events of a series

The recurring appointments can be edited from current and following events when enable the property `editFollowingEvents`.

Editing Following Events - When you double click on a recurring event, a popup prompts you to choose either to edit the single event or Edit Following Events or entire series. From this, if you choose to select **EDIT FOLLOWING EVENTS** option, a current and following events of the recurring appointment will be edited. The following process takes place while editing a following events,

- A new event will be created from the parent event data and added to the Scheduler dataSource, with all its default field values overwritten with the newly modified data and additionally, the `followingID` field will be added to it, that holds the `id` value of the immediate parent recurring event. Also, a new `Id` will be generated for this event in the dataSource.
- The parent recurring event needs to be updated with appropriate `recurrenceRule` field to hold the modified occurrence appointment's end date.

Therefore, when a following events are edited from a recurring event, the batch action takes place by allowing the `Add`, `Edit` and `Delete` action requests to take place together.

```
`ts
```

```

if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
will execute while inserting the appointments
{
    var value = (param.action == "insert") ? param.value : param.added[0];
    int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
    1;
    DateTime startTime = Convert.ToDateTime(value.StartTime);
    DateTime endTime = Convert.ToDateTime(value.EndTime);
    ScheduleEventData appointment = new ScheduleEventData()
    {

```

```
Id = intMax + 1,
StartTime = startTime.ToLocalTime(),
EndTime = endTime.ToLocalTime(),
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
FollowingID = value.FollowingID,
RecurrenceID = value.RecurrenceID,
RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
```

```

appointment.FollowingID = value.FollowingID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`

```

Restricting edit action based on specific criteria

You can also dynamically prevent the editing of appointments on Scheduler. For example, say if you want to decline the updating of appointments on non-working hours, you can check for its appropriate condition within the `actionBegin` event.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda, ActionEventArgs }
from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',

```

```

selectedDate: new Date(2018, 1, 15),
eventSettings: { dataSource: scheduleData },
actionBegin: (args: ActionEventArgs) => {
    if (args.requestType == 'eventChange') {
        let weekEnds: number[] = [0, 6];
        let data: { [key: string]: Object } = args.data as { [key: string]:
Object };
        let weekDay: boolean = weekEnds.indexOf((data.StartTime as
Date).getDay()) >= 0;
        let workHours: boolean =
((parseInt(scheduleObj.workHours.start.toString().slice(0, 2), 10) <=
(data.StartTime as Date).getHours()) &&
(parseInt(scheduleObj.workHours.end.toString().slice(0, 2), 10)) >=
(data.StartTime as Date).getHours());
        if (weekDay || !workHours) {
            args.cancel = true;
        }
    }
}
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Delete

The appointments can be deleted in either of the following ways,

- Selecting an appointment and clicking the delete icon from the quick popup that opens.
- Selecting an appointment and pressing **Delete** key.
- Selecting multiple appointments by tap holding an event and then continuously single clicking on other consecutive events and then clicking the **Delete** key.
- Double clicking on an event which opens the default event editor pre-filled with event details, and then choosing **Delete** button in it.

While performing all these above mentioned actions, a pop-up with a delete confirmation message will be displayed prompting either to proceed with deleting an appointment.

Deletion using editor window

When you double click an event, the default editor window will be opened which includes a **Delete** button at the bottom left position to allow you to delete that particular appointment. When deleting an appointment through this editor window, the delete alert confirmation will not be asked and the event will be deleted immediately.

Deletion using deleteEvent method

The appointments can be removed manually using the **deleteEvent** method. The following code examples shows how to edit the normal and recurring events.

Normal event - You can delete the normal appointments of Scheduler by simply passing its **Id** value or the entire event object collection to the **deleteEvent** method.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleData: Object[] = [{
    Id: 3,
    Subject: 'Testing',
    StartTime: new Date(2018, 1, 11, 9, 0),
    EndTime: new Date(2018, 1, 11, 10, 0),
    IsAllDay: false
}, {

```

```

        Id: 4,
        Subject: 'Vacation',
        StartTime: new Date(2018, 1, 13, 9, 0),
        EndTime: new Date(2018, 1, 13, 10, 0),
        IsAllDay: false
    }];
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: {
        dataSource: scheduleData
    }
});
scheduleObj.appendTo('#Schedule');
let remove: Button = new Button();
remove.appendTo('#delete');
remove.element.onclick = (): void => {
    scheduleObj.deleteEvent(4);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <button id="delete">Delete</button>
  <div id="Schedule"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Recurring Event - The recurring events can be removed as an entire series or simply removing single occurrence by using the deleteEvent method which takes in either the DeleteSeries or DeleteOccurrence parameters. The following code example shows how to delete entire series.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleData: Object[] = [{
  Id: 3,
  Subject: 'Testing',
  StartTime: new Date(2018, 1, 11, 9, 0),
  EndTime: new Date(2018, 1, 11, 10, 0),
  IsAllDay: false,
  RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3'
}, {
  Id: 4,
  Subject: 'Vacation',
  StartTime: new Date(2018, 1, 12, 11, 0),
  EndTime: new Date(2018, 1, 12, 12, 0),
  IsAllDay: false,
  RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
}];
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'WorkWeek', 'Month'],
  eventSettings: {
    dataSource: scheduleData
  }
});
scheduleObj.appendTo('#Schedule');
let remove: Button = new Button();
remove.appendTo('#delete');
remove.element.onclick = (): void => {
  let scheduleData: { [key: string]: Object }[] = [{
    Id: 4,
    Subject: 'Vacation',
    RecurrenceID: 4,
    StartTime: new Date(2018, 1, 12, 11, 0),
    EndTime: new Date(2018, 1, 12, 12, 0),

```

```

        IsAllDay: false,
        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
    }];
    scheduleObj.deleteEvent(scheduleData, 'DeleteSeries');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

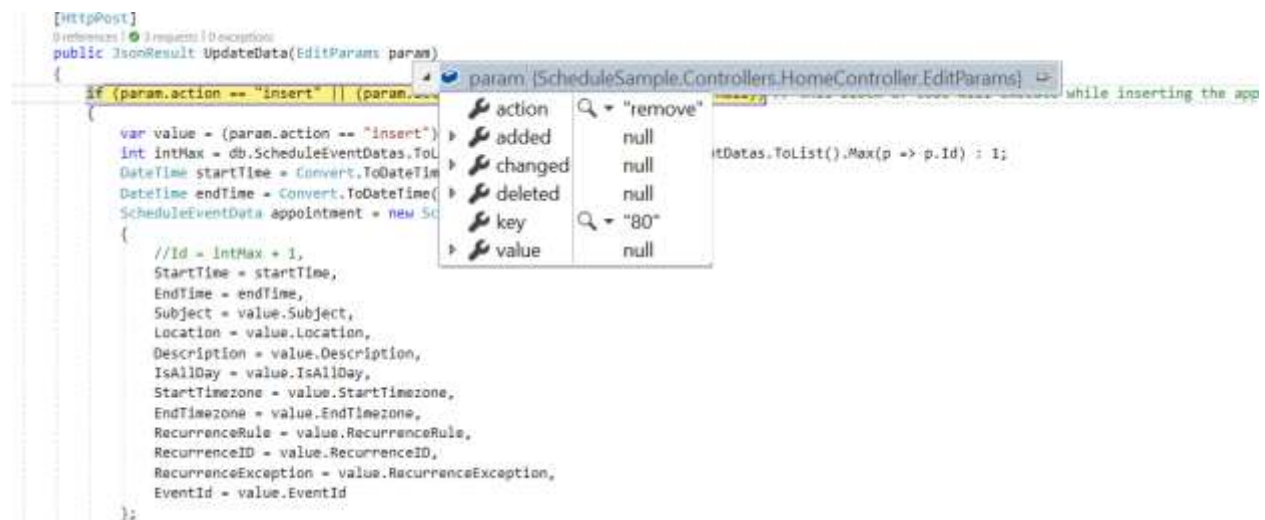
    <div id="container">
        <button id="delete">Delete</button>
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Removing events from database at server-side

While deleting the event from the Scheduler, **remove** action takes place and the following code example describes how to delete event from database at server side.

```
`ts
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
```



How to delete a single occurrence or entire series from Scheduler and update it in database at server-side

The recurring events can be deleted in either of the following two ways.

- Single occurrence
- Entire series

Single occurrence - When you attempt to delete the recurring events, a popup prompts you to choose either to delete the single event or entire series. From this, if you choose to select **DELETE EVENT** option, a single occurrence of the recurring appointment alone will be removed. The following process takes place while removing a single occurrence,

- The selected occurrence will be deleted from the Scheduler user interface.
- In code, the parent recurring event object will be updated with appropriate `recurrenceException` field, to hold the deleted occurrence appointment's date collection.

Therefore, when a single occurrence is deleted from a recurring event, the `update` action takes place on the parent recurring event as shown in the following code example.

In case, if you delete an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get removed. In this case, `delete` action takes place instead of `update` action and the parent recurring event object remains same with no changes.

```
`ts
```

```
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
```

```
{
```

```
var value = (param.action == "update") ? param.value : param.changed[0];
```

```
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
```

```
if (filterData.Count() > 0)
```

```
{
```

```
DateTime startTime = Convert.ToDateTime(value.StartTime);
```

```
DateTime endTime = Convert.ToDateTime(value.EndTime);
```

```
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
```

```
appointment.StartTime = startTime.ToLocalTime();
```

```
appointment.EndTime = endTime.ToLocalTime();
```

```
appointment.StartTimezone = value.StartTimezone;
```

```
appointment.EndTimezone = value.EndTimezone;
```

```
appointment.Subject = value.Subject;
```

```
appointment.IsAllDay = value.IsAllDay;
```

```

appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
`ts

```

Entire series - When you select an option **DELETE SERIES** from the popup, the whole recurring series will be deleted. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences which are maintained as separate event objects will also be removed from the dataSource. This action of deleting entire series leads to **remove** action and removes one or more event objects at the same time.

```

`ts
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`ts

```

How to delete only the current and following events of a series

The recurring events can be deleted from current and following events only when enable `editFollowingEvents` property.

Delete Following Events - When you attempt to delete the recurring events, a popup prompts you to choose either to delete the single event or Following Events or entire series. From this, if you choose to select **FOLLOWING EVENT** option, a current and following events of the recurring appointment alone will be removed. The following process takes place while removing a single occurrence,

- The selected occurrence and the following events in same series will be deleted from the Scheduler user interface.
- In code, the parent recurring event object will be updated with appropriate `recurrenceRule` field, to update the end date of the recurring events.

Therefore, when following events are deleted from a recurring event, the `remove` and `update` action takes place on the immediate parent recurring event as shown in the following code example.

```
`ts
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime.ToLocalTime();
appointment.EndTime = endTime.ToLocalTime();
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.followingID = value.followingID;
appointment.RecurrenceException = value.RecurrenceException;
}
}
```

```

db.SubmitChanges();
}

if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`

```

Drag and drop

When you drag and drop a normal event on the Scheduler, the event editing action takes place. When a recurring event is drag and dropped on a desired time range, the batch action explained in [Editing a single occurrence](#) process will takes place - thus allowing both the [Add](#) and [Edit](#) action to take place together.

By default, when you drag a recurring instance, only the occurrence of the event gets edited and not a whole series.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, DragAndDrop } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from '../datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, DragAndDrop);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',

```

```

        selectedDate: new Date(2018, 1, 15),
        eventSettings: { dataSource: data }
    });
    scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```


Resize

When you resize a normal event on the Scheduler, the event editing action takes place. When a recurring event is resized to a new desired time, the batch action explained in [Editing a single occurrence](#) process will take place - thus allowing both the **Add** and **Edit** action to take place together.

By default, when you resize a recurring instance, only the occurrence of the event gets edited and not a whole series.

INDEX.TS

```
import { extend } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, Resize } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Resize);
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: data }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```

<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Virtual scrolling in EJ2 JavaScript Schedule control

To achieve better performance in the Scheduler when loading a large number of resources and events, we have added virtual scrolling support to load a large set of resources and events instantly as you scroll. You can dynamically load large number of resources and events in the Scheduler by setting `true` to the [allowVirtualScrolling](#) property within the view specific settings. The virtual loading of events is possible in Agenda view, by setting [allowVirtualScrolling](#) property to `true` within the agenda view specific settings.

INDEX.TS

```

import { Schedule, TimelineViews, TimelineMonth, TimelineYear, Resize,
DragAndDrop } from '@syncfusion/ej2-schedule';
Schedule.Inject(TimelineViews, TimelineMonth, TimelineYear, Resize,
DragAndDrop);
let ownerData: Object[] = generateResourceData(1, 300, 'Resource');
let eventData: Object[] = generateStaticEvents(new Date(2018, 4, 1), 300,
12);
let scheduleObj: Schedule = new Schedule({
  height: '550px', width: '100%',
  currentView: 'TimelineMonth',
  views: [
    { option: 'TimelineMonth', eventTemplate: '#timeline-event-
template', allowVirtualScrolling: true },
    { option: 'TimelineYear', orientation: 'Vertical', eventTemplate:
'#timeline-event-template', allowVirtualScrolling: true }
  ],
  group: {
    byGroupID: false,
    resources: ['Owners']
  },
  resources: [
    {
      field: 'OwnerId', title: 'Owner',
      name: 'Owners', allowMultiple: true,
      dataSource: ownerData,
      textField: 'Text', idField: 'Id', colorField: 'Color'
    }
  ]
});

```

```

    }
    ],
    selectedDate: new Date(2018, 4, 1),
    eventSettings: { dataSource: eventData }
});
scheduleObj.appendTo('#Schedule');
function generateStaticEvents(start: Date, resCount: number, overlapCount:
number): Object[] {
    let data: Object[] = [];
    let id: number = 1;
    for (let i: number = 0; i < resCount; i++) {
        let randomCollection: number[] = [];
        let random: number = 0;
        for (let j: number = 0; j < overlapCount; j++) {
            random = Math.floor(Math.random() * (30));
            random = (random === 0) ? 1 : random;
            if (randomCollection.indexOf(random) !== -1 ||
randomCollection.indexOf(random + 2) !== -1 ||
randomCollection.indexOf(random - 2) !== -1) {
                random += (Math.max.apply(null, randomCollection) + 10);
            }
            for (let k: number = 1; k <= 2; k++) {
                randomCollection.push(random + k);
            }
            let startDate: Date = new Date(start.getFullYear(),
start.getMonth(), random);
            startDate = new Date(startDate.getTime() + (((random % 10) * 10)
* (1000 * 60)));
            let endDate: Date = new Date(startDate.getTime() + ((1440 + 30)
* (1000 * 60)));
            data.push({
                Id: id,
                Subject: 'Event #' + id,
                StartTime: startDate,
                EndTime: endDate,
                IsAllDay: (id % 10) ? false : true,
                OwnerId: i + 1
            });
            id++;
        }
    }
    return data;
}
function generateResourceData(startId: number, endId: number, text: string):
Object[] {
    let data: { [key: string]: Object }[] = [];
    let colors: string[] = [
        '#ff8787', '#9775fa', '#748ffc', '#3bc9db', '#69db7c',
        '#fdd835', '#748ffc', '#9775fa', '#df5286', '#7fa900',
        '#fec200', '#5978ee', '#00bdae', '#ea80fc'
    ];
    for (let a: number = startId; a <= endId; a++) {
        let n: number = Math.floor(Math.random() * colors.length);
        data.push({
            Id: a,
            Text: text + ' ' + a,
            Color: colors[n]
        });
    }
}

```

```

    });
}
return data;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-schedule .e-timeline-month-view .template-wrap .subject {
      padding: 10px 25px;
    }

    .e-schedule .template-wrap {
      width: 100%;
    }

    .e-schedule .e-timeline-year-view .template-wrap .subject {
      padding: 1px 25px;
    }

    .e-schedule .e-more-event-popup .template-wrap .subject {
      padding: 0px 25px;
    }

    .e-schedule .e-timeline-month-view .e-resource-left-td {
      width: 150px;
    }
  </style>
  <script id="timeline-event-template" type="text/x-template">
    <div class='template-wrap' style='background:${PrimaryColor}'>
      <div class="subject"
style='background:${SecondaryColor};'>${Subject}</div>
    </div>
  </script>

```

```

</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: For now, the virtual loading of resources and events is not supported in **MonthAgenda**, **Year** and **TimelineYear** (Horizontal Orientation) views.

Enabling lazy loading for appointments

The lazy loading feature provides a convenient way to efficiently load resource appointments into the Scheduler using an on-demand approach. With this feature, you can seamlessly load a large volume of appointment data into the Scheduler without experiencing any performance degradation.

By default, the Scheduler fetches all the relevant appointments from the server with in the current date range. However, enabling this feature will trigger query requests to the server for appointment retrieval whenever new resources are rendered due to scroll actions. These queries contain the resource IDs of currently displayed resources along with current date range, which can be passed as a comma-separated string. In the server controller, these resource IDs are parsed to filter the necessary appointments to render in the scheduler.

When you enable this feature, the Scheduler becomes capable of fetching events from remote services only for the current view port alone to optimize the data retrieval. The remaining appointment data is fetched form the server on-demand based on currently rendered view port resources as you scroll's through the scheduler content.

To enable this feature, you have to set the [enableLazyLoading](#) property to **true** within the view specific settings.

INDEX.TS

```

import { Schedule, TimelineMonth } from '@syncfusion/ej2-schedule';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
Schedule.Inject(TimelineMonth);
let resourceData: Object[] = generateResourceData(1, 1000, 'Resource');
let dataManager: DataManager = new DataManager({
    url:
    'https://services.syncfusion.com/js/production/api/VirtualEventData',
    adaptor: new WebApiAdaptor,
    crossDomain: true
});

```

```

let scheduleObj: Schedule = new Schedule({
  height: '550px', width: '100%',
  currentView: 'TimelineMonth',
  readonly: true,
  views: [
    { option: 'TimelineMonth', enableLazyLoading: true }
  ],
  group: {
    resources: ['Resources']
  },
  resources: [
    {
      field: 'ResourceId', title: 'Resources',
      name: 'Resources',
      dataSource: resourceData,
      textField: 'Text', idField: 'Id', colorField: 'Color'
    }
  ],
  selectedDate: new Date(2023, 3, 1),
  eventSettings: { dataSource: dataManager }
});
scheduleObj.appendTo('#Schedule');
function generateResourceData(startId: number, endId: number, text: string):
Object[] {
  let data: { [key: string]: Object }[] = [];
  let colors: string[] = [
    '#ff8787', '#9775fa', '#748ffc', '#3bc9db', '#69db7c',
    '#fdd835', '#748ffc', '#9775fa', '#df5286', '#7fa900',
    '#fec200', '#5978ee', '#00bdae', '#ea80fc'
  ];
  for (let a: number = startId; a <= endId; a++) {
    let n: number = Math.floor(Math.random() * colors.length);
    data.push({
      Id: a,
      Text: text + ' ' + a,
      Color: colors[n]
    });
  }
  return data;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Here's the server-side controller code that retrieves appointment data based on the resource IDs provided as query parameters:

```

`c#
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.OData.Query;
namespace LazyLoadingServices.Controllers
{
    public class VirtualEventDataController : Controller
    {
        private readonly EventsContext dbContext;

        [HttpGet]

```

```
[EnableQuery]
[Route("api/VirtualEventData")]
public IActionResult GetData([FromQuery] Params param)
{
    IQueryable<EventData> query = dbContext.Events;
    // Filter the appointment data based on the ResourceId query params.
    if (!string.IsNullOrEmpty(param.ResourceId))
    {
        string[] resourceId = param.ResourceId.Split(',');
        query = query.Where(data => resourceId.Contains(data.ResourceId.ToString()));
    }
    return Ok(query.ToList());
}
}
public class Params
{
    public DateTime? StartDate { get; set; }
    public DateTime? EndDate { get; set; }
    public string ResourceId { get; set; }
}
`
```

Note:

- The property will be effective, when large number of resources and appointments bound to the Scheduler.
- This property is applicable only when [resource grouping](#) is enabled in Scheduler.

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

See Also

- [Virtual scrolling in Agenda view](#)

Editor template in EJ2 JavaScript Schedule control

Scheduler makes use of popups and dialog to display the required notifications, as well as includes an editor window with event fields for making the appointment creation and editing process easier. You can also easily customize the editor window and the fields present in it, and can also apply validations on those fields.

Event editor

The editor window usually opens on the Scheduler, when a cell or event is double clicked. When a cell is double clicked, the detailed editor window opens in "Add new" mode, whereas when an event is double clicked, the same is opened in an "Edit" mode.

In mobile devices, you can open the detailed editor window in edit mode by clicking the edit icon on the popup, that opens on single tapping an event. You can also open it in add mode by single tapping a cell, which will display a + indication, clicking on it again will open the editor window.

You can also prevent the editor window from opening, by rendering Scheduler in a **readonly** mode or by doing code customization within the **popupOpen** event.

How to change the editor window header title and text of footer buttons

You can change the header title and the text of buttons displayed at the footer of the editor window by changing the appropriate localized word collection used in the Scheduler.

INDEX.TS

```
import { L10n } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, TimelineViews, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda);
L10n.load({
  'en-US': {
    'schedule': {
      'saveButton': 'Add',
      'cancelButton': 'Close',
      'deleteButton': 'Remove',
      'newEvent': 'Add Event',
    },
  },
});
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['TimelineDay', 'Day', 'Week', 'Month', 'Agenda'],
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-textlabel {
        font-weight: bold;
        padding-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to change the label text of default editor fields

To change the default labels such as Subject, Location and other field names in the editor window, make use of the `title` property available within the field option of `eventSettings`.

INDEX.TS

```

import { Schedule, Day, Week, TimelineViews, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';

```

```

Schedule.Inject(Day, Week, TimelineViews, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
  eventSettings: {
    dataSource: scheduleData,
    fields: {
      id: 'Id',
      subject: { name: 'Subject', title: 'Event Name' },
      location: { name: 'Location', title: 'Event Location' },
      description: { name: 'Description', title: 'Event Description' }
    },
    startTime: { name: 'StartTime', title: 'Start Duration' },
    endTime: { name: 'EndTime', title: 'End Duration' }
  }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-textlabel {
      font-weight: bold;
      padding-right: 5px;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Field validation

It is possible to validate the required fields of the editor window from client-side before submitting it, by adding appropriate validation rules to each field. The appointment fields have been extended to accept both **string** and **object** type values. To perform validations, it is necessary to specify object values for the event fields.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let minValidation: (args: { [key: string]: string }) => boolean = (args: {
[key: string]: string }) => {
    return args['value'].length >= 5;
};
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '500px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: {
        dataSource: scheduleData,
        fields: {
            id: 'Id',
            subject: { name: 'Subject', validation: { required: true } },
            location: { name: 'Location', validation: { required: true } },
            description: {
                name: 'Description', validation: {
                    required: true, minLength: [minValidation, 'Need atleast
5 letters to be entered']
                }
            },
            startTime: { name: 'StartTime', validation: { required: true } },
            endTime: { name: 'EndTime', validation: { required: true } }
        }
    }
});

```

```
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-textlabel {
      font-weight: bold;
      padding-right: 5px;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="button"></div>
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Applicable validation rules can be referred from [form validation](#) documentation.

Add additional fields to the default editor

The additional fields can be added to the default event editor by making use of the `popupOpen` event which gets triggered before the event editor opens on the Scheduler. The `popupOpen` is a client-side event that triggers before any of the generic popups opens on the Scheduler. The additional field (any of the form elements) should be added with a common class name `e-field`, so as to handle and process those additional data along with the default event object. In the following example, an additional field `Event Type` has been added to the default event editor and its value is processed accordingly.

INDEX.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, PopupOpenEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'Editor') {
            // Create required custom elements in initial time
            if (!args.element.querySelector('.custom-field-row')) {
                let row: HTMLElement = createElement('div', { className: 'custom-field-row' });
                let formElement: HTMLElement = args.element.querySelector('.e-schedule-form');
                formElement.firstChild.insertBefore(row, args.element.querySelector('.e-title-location-row'));
                let container: HTMLElement = createElement('div', { className: 'custom-field-container' });
                let inputEle: HTMLInputElement = createElement('input', {
                    className: 'e-field',
                    attrs: { name: 'EventType' }
                }) as HTMLInputElement;
                container.appendChild(inputEle);
                row.appendChild(container);
                let dropDownList: DropDownList = new DropDownList({
                    dataSource: [
                        { text: 'Public Event', value: 'public-event' },
                        { text: 'Maintenance', value: 'maintenance' },
                        { text: 'Commercial Event', value: 'commercial-event' },
                        { text: 'Family Event', value: 'family-event' }
                    ],
                    fields: { text: 'text', value: 'value' },
                    value: (<{ [key: string]: Object }>(args.data)).EventType as string,
                    floatLabelType: 'Always',
                    placeholder: 'Event Type'
                });
                dropDownList.appendTo(inputEle);
                inputEle.setAttribute('name', 'EventType');
```

```

    }
  },
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-textlabel {
      font-weight: bold;
      padding-right: 5px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="button"></div>
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the default time duration in editor window

In default event editor window, start and end time duration are processed based on the **interval** value set within the **timeScale** property. By default, **interval** value is set to 30, and therefore the start/end time duration within the event editor will be in a 30 minutes time difference. You can change this duration value by changing the **duration** option within the **popupOpen** event as shown in the following code example.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, PopupOpenEventArgs } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'WorkWeek', 'Month'],
  popupOpen: (args: PopupOpenEventArgs) => {
    if (args.type === 'Editor') {
      args.duration = 60;
    }
  },
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-textlabel {
        font-weight: bold;
        padding-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to prevent the display of editor and quick popups

It is possible to prevent the display of editor and quick popup windows by passing the value `true` to `cancel` option within the `popupOpen` event.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, PopupOpenEventArgs } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    popupOpen: (args: PopupOpenEventArgs) => {
        args.cancel = true;
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-textlabel {
      font-weight: bold;
      padding-right: 5px;
    }
  </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="button"></div>
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the `popupOpen` event are as follows.

Type	Description
----- -----	
Editor	For Detailed editor window.
QuickInfo	For Quick popup which opens on cell click.
EditEventInfo	For Quick popup which opens on event click.
ViewEventInfo	For Quick popup which opens on responsive mode.
EventContainer	For more event indicator popup.
RecurrenceAlert	For edit recurrence event alert popup.
DeleteAlert	For delete confirmation popup.
ValidationAlert	For validation alert popup.
RecurrenceValidationAlert	For recurrence validation alert popup.

Customizing timezone collection in the editor window

By default, the timezone collections in the editor window have been loaded with built-in timezone collections. Now we can be able to customize the timezone collections using the `timezoneDataSource` property with the collection of `TimezoneFields` data.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object[] = [{
  Id: 1,
  Subject: 'Explosion of Betelgeuse Star',
  StartTime: new Date(2020, 1, 15, 10, 0),
  EndTime: new Date(2018, 1, 15, 12, 30),
  IsAllDay: false
}, {
  Id: 2,
  Subject: 'Blue Moon Eclipse',
  StartTime: new Date(2020, 1, 16, 12, 0),
  EndTime: new Date(2018, 1, 16, 13, 0),
  IsAllDay: false
}];
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: { dataSource: data },
  timezoneDataSource: [
    { Value: 'Pacific/Niue', Text: 'Niue' },
    { Value: 'Pacific/Pago_Pago', Text: 'Pago Pago' },
    { Value: 'Pacific/Honolulu', Text: 'Hawaii Time' },
    { Value: 'Pacific/Rarotonga', Text: 'Rarotonga' },
    { Value: 'Pacific/Tahiti', Text: 'Tahiti' },
  ],
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-textlabel {
      font-weight: bold;
      padding-right: 5px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="button"></div>
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing event editor using template

The event editor window can be customized by making use of the `editorTemplate` option. Here, the custom window design is built with the required fields using the script template and its type should be of `text/x-template`.

Each field defined within template should contain the **e-field** class, so as to allow the processing of those field values internally. The ID of this customized script template section is assigned to the `editorTemplate` option, so that these customized fields will be replaced onto the default editor window.

Note: **e-field** class only applicable for **DropDownList**, **DateTimePicker**, **MultiSelect**, **DatePicker**, **CheckBox** and **TextBox** components. Since we have processed the field values internally for the above mentioned components.

As we are using our Syncfusion sub-components within our editor using template in the following example, the custom defined form elements needs to be configured as required Syncfusion components such as **DropDownList** and **DateTimePicker** within the `popupOpen` event. This particular step can be skipped, if the user needs to simply use the usual form elements.

INDEX.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, PopupOpenEventArgs }
from '@syncfusion/ej2-schedule';
import { eventData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  showQuickInfo: false,
  editorTemplate: '#EventEditorTemplate',
  popupOpen: (args: PopupOpenEventArgs) => {
    if (args.type === 'Editor') {
      let statusElement: HTMLInputElement =
args.element.querySelector('#EventType') as HTMLInputElement;
      if (!statusElement.classList.contains('e-dropdownlist')) {
        let dropDownListObject: DropDownList = new
DropDownList({
          placeholder: 'Choose status', value:
statusElement.value,
          dataSource: ['New', 'Requested', 'Confirmed']
        });
        dropDownListObject.appendTo(statusElement);
      }
      let startElement: HTMLInputElement =
args.element.querySelector('#StartTime') as HTMLInputElement;
      if (!startElement.classList.contains('e-datetimepicker')) {
        new DateTimePicker({ value: new Date(startElement.value)
|| new Date(), startElement);
      }
      let endElement: HTMLInputElement =
args.element.querySelector('#EndTime') as HTMLInputElement;
      if (!endElement.classList.contains('e-datetimepicker')) {
```

```

        new DateTimePicker({ value: new Date(endElement.value)
|| new Date() }, endElement);
    }
    },
    eventSettings: { dataSource: eventData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .custom-event-editor .e-textlabel {
      padding-right: 15px;
      text-align: right;
    }
    .custom-event-editor td {
      padding: 7px;
      padding-right: 16px;
    }
  </style>
  <script id="EventEditorTemplate" type="text/x-template">
    <table class="custom-event-editor" width="100%" cellpadding="5">
      <tbody>
        <tr>
          <td class="e-textlabel">Summary</td>
          <td colspan="4">
            <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
          </td>
        </tr>
      </tbody>
    </table>
  </script>

```

```

        <tr>
            <td class="e-textlabel">Status</td>
            <td colspan="4">
                <input type="text" id="EventType" name="EventType"
class="e-field" style="width: 100%" />
            </td>
        </tr>
        <tr>
            <td class="e-textlabel">From</td>
            <td colspan="4">
                <input id="StartTime" class="e-field" type="text"
name="StartTime" />
            </td>
        </tr>
        <tr>
            <td class="e-textlabel">To</td>
            <td colspan="4">
                <input id="EndTime" class="e-field" type="text"
name="EndTime" />
            </td>
        </tr>
        <tr>
            <td class="e-textlabel">Reason</td>
            <td colspan="4">
                <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50"
style="width: 100%; height: 60px !important; resize:
vertical"></textarea>
            </td>
        </tr>
    </tbody>
</table>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to customize header and footer using template

The editor window's header and footer can be enhanced with custom designs using the [editorHeaderTemplate](#) and [editorFooterTemplate](#) options. To achieve this, create a script template that includes the necessary fields. Ensure that the template type is set to **text/x-template**.

In this demo, we tailor the editor's header according to the appointment's subject field using the [editorHeaderTemplate](#). Furthermore, we make use of the [editorFooterTemplate](#) to handle the functionality of validating specific fields before proceeding with the save action or canceling it if validation requirements are not met.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda, PopupOpenEventArgs }
from '@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
const today: Date = new Date();
const data : Record<string, any>[] = [{
    Id: 1,
    Subject: 'Surgery - Andrew',
    StartTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 9, 0),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 10, 0),
    IsAllDay: false
}, {
    Id: 2,
    Subject: 'Consulting - John',
    StartTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 10, 0),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 11, 30),
    IsAllDay: false
}, {
    Id: 3,
    Subject: 'Therapy - Robert',
    StartTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 11, 30),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 12, 30),
    IsAllDay: false
}, {
    Id: 4,
    Subject: 'Observation - Steven',
    StartTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 12, 30),
    EndTime: new Date(today.getFullYear(), today.getMonth(),
today.getDate(), 13, 30),
    IsAllDay: false
}]
const scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    editorFooterTemplate: '#editor-footer',
    editorHeaderTemplate: '#editor-header',
    eventSettings: { dataSource: data },
    popupOpen: onPopupOpen
```



```

});
scheduleObj.appendTo('#Schedule');
function onSaveButtonClick(args: PopupOpenEventArgs) {
    const data: Record<string, any> = {
        Id: args.data.Id,
        Subject: (args.element.querySelector('#Subject') as
HTMLInputElement).value,
        StartTime: (args.element.querySelector('#StartTime') as
any).ej2_instances[0].value,
        EndTime: (args.element.querySelector('#EndTime') as
any).ej2_instances[0].value,
        IsAllDay: (args.element.querySelector('#IsAllDay') as
HTMLInputElement).checked
    };
    if (args.target.classList.contains('e-appointment')) {
        scheduleObj.saveEvent(data, 'Save');
    } else {
        data.Id = scheduleObj.getEventMaxID();
        scheduleObj.addEvent(data);
    }
    scheduleObj.closeEditor();
}
function onPopupOpen(args: PopupOpenEventArgs): void {
    if (args.type === 'Editor') {
        const saveButton: HTMLElement = args.element.querySelector('#Save')
as HTMLElement;
        const cancelButton: HTMLElement =
args.element.querySelector('#Cancel') as HTMLElement;
        const checkBox: HTMLInputElement =
args.element.querySelector('#check-box') as HTMLInputElement;
        checkBox.onchange = () => {
            if (!(checkBox as HTMLInputElement).checked) {
                saveButton.setAttribute('disabled', '');
            } else {
                saveButton.removeAttribute('disabled');
            }
        };
        saveButton.onclick = () => {
            onSaveButtonClick(args);
        }
        cancelButton.onclick = () => {
            scheduleObj.closeEditor();
        };
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<script id="editor-header" type="text/x-template">
    ${if (!Subject)}<div>Create New
Event</div>${else}<div>${Subject}</div>${/if}
</script>
<script id="editor-footer" type="text/x-template">
    <div id="verify">
        <input type="checkbox" id="check-box" value="unchecked">
        <label id="text">Verified</label>
    </div>
    <div id="right-button">
        <button id="Save" class="e-control e-btn e-primary" disabled data-
ripple="true">Save</button>
        <button id="Cancel" class="e-control e-btn e-primary" data-
ripple="true">Cancel</button>
    </div>
</script>
<style>
    #verify {
        position: fixed;
        padding: 0 20px;
    }
    #text {
        cursor: pointer;
        display: inline-block;
        font-family: "Roboto", -apple-system, BlinkMacSystemFont, "Segoe
UI", "Helvetica Neue", sans-serif;
        font-size: 14px;
        font-weight: normal;
        line-height: 14px;
        user-select: none;
        margin-left: 8px;
        vertical-align: middle;
        white-space: normal;
    }
    #right-button {

```

```

        padding: 0 10px;
    }
</style>
<body>
    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
        var ele = document.getElementById('container');
        if (ele) {
            ele.style.visibility = "visible";
        }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

How to add resource options within editor template

The resource field can be added within editor template with multiselect control for allow multiple resources.

INDEX.TS

```

import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { MultiSelect } from '@syncfusion/ej2-dropdowns';
import { Schedule, Day, Week, WorkWeek, Month, PopupOpenEventArgs } from '@syncfusion/ej2-schedule';
import { eventData, ownerData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    group: { resources: ['Owners'] },
    showQuickInfo: false,
    resources: [{
        field: 'OwnerId', title: 'Owners',
        name: 'Owners', allowMultiple: true,
        dataSource: [
            { text: "Nancy", id: 1, color: "#1aaa55" },
            { text: "Smith", id: 2, color: "#7fa900" },
            { text: "Paul", id: 3, color: "#357cd2" }
        ],
        textField: 'text', idField: 'id', colorField: 'color'
    }],
    editorTemplate: '#EventEditorTemplate',
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'Editor') {
            let startElement: HTMLInputElement =
args.element.querySelector('#StartTime') as HTMLInputElement;
            if (!startElement.classList.contains('e-datetimepicker')) {
                new DateTimePicker({ value: new Date(startElement.value) ||
new Date() }, startElement);
            }
        }
    }
});

```

```

        let endElement: HTMLInputElement =
args.element.querySelector('#EndTime') as HTMLInputElement;
        if (!endElement.classList.contains('e-datetimepicker')) {
            new DateTimePicker({ value: new Date(endElement.value) ||
new Date() }, endElement);
        }
        let processElement: HTMLInputElement=
args.element.querySelector('#OwnerId');
        if (!processElement.classList.contains('e-multiselect')) {
            let multiSelectObject: MultiSelect = new MultiSelect({
                placeholder: 'Choose a owner',
                fields: { text: 'text', value: 'id'},
                dataSource: ownerData,
                value: <string[]>((args.data.OwnerId instanceof Array) ?
args.data.OwnerId : [args.data.OwnerId])
            });
            multiSelectObject.appendTo(processElement);
        }
    },
    eventSettings: { dataSource: eventData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .custom-event-editor .e-textlabel {
            padding-right: 15px;
            text-align: right;
        }
    </style>

```

```

        .custom-event-editor td {
            padding: 7px;
            padding-right: 16px;
        }
    </style>
    <script id="EventEditorTemplate" type="text/x-template">
        <table class="custom-event-editor" width="100%" cellpadding="5">
            <tbody>
                <tr>
                    <td class="e-textlabel">Summary</td>
                    <td colspan="4">
                        <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
                    </td>
                </tr>
                <tr>
                    <td class="e-textlabel">From</td>
                    <td colspan="4">
                        <input id="StartTime" class="e-field" type="text"
name="StartTime" />
                    </td>
                </tr>
                <tr>
                    <td class="e-textlabel">To</td>
                    <td colspan="4">
                        <input id="EndTime" class="e-field" type="text"
name="EndTime" />
                    </td>
                </tr>
                <tr>
                    <td class="e-textlabel">Owner</td>
                    <td colspan="4">
                        <input type="text" id="OwnerId" name="OwnerId" class="e-
field" style="width: 100%" />
                    </td>
                </tr>
                <tr>
                    <td class="e-textlabel">Reason</td>
                    <td colspan="4">
                        <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50"
style="width: 100%; height: 60px !important; resize:
vertical"></textarea>
                    </td>
                </tr>
            </tbody>
        </table>
    </script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to add recurrence options within editor template

The following code example shows how to add recurrence options within the editor template by importing `RecurrenceEditor`.

INDEX.TS

```

import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { Schedule, Day, Week, WorkWeek, Month, RecurrenceEditor,
PopupOpenEventArgs } from '@syncfusion/ej2-schedule';
import { eventData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    editorTemplate: '#EventEditorTemplate',
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'Editor') {
            let startElement: HTMLInputElement =
args.element.querySelector('#StartTime') as HTMLInputElement;
            if (!startElement.classList.contains('e-datetimepicker')) {
                new DateTimePicker({ value: new Date(startElement.value) ||
new Date() }, startElement);
            }
            let endElement: HTMLInputElement =
args.element.querySelector('#EndTime') as HTMLInputElement;
            if (!endElement.classList.contains('e-datetimepicker')) {
                new DateTimePicker({ value: new Date(endElement.value) ||
new Date() }, endElement);
            }
            let recurElement: HTMLElement =
args.element.querySelector('#RecurrenceEditor');
            if (!recurElement.classList.contains('e-recurrenceeditor')) {
                let recurrObject: RecurrenceEditor = new RecurrenceEditor({
                });
                recurrObject.appendTo(recurElement);
                (scheduleObj.eventWindow as any).recurrenceEditor =
recurrObject;
            }
            document.getElementById('RecurrenceEditor').style.display =
(scheduleObj.currentAction === "EditOccurrence") ? 'none' : 'block';
        }
    },
    eventSettings: { dataSource: eventData }
}

```

```
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .custom-event-editor .e-textlabel {
      padding-right: 15px;
    }
    .custom-event-editor td {
      padding: 7px;
    }
  </style>
<script id="EventEditorTemplate" type="text/x-template">
  <table class="custom-event-editor" width="100%" cellpadding="5">
    <tbody>
      <tr>
        <td class="e-textlabel">Summary</td>
        <td colspan="4">
          <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
        </td>
      </tr>
      <tr>
        <td class="e-textlabel">From</td>
        <td colspan="4">
          <input id="StartTime" class="e-field" type="text"
name="StartTime" />
        </td>
      </tr>
    </tbody>
  </table>
```

```

        <td class="e-textlabel">To</td>
        <td colspan="4">
            <input id="EndTime" class="e-field" type="text"
name="EndTime" />
        </td>
    </tr>
    <tr>
        <td colspan="4">
            <div id='RecurrenceEditor'></div>
        </td>
    </tr>
    <tr>
        <td class="e-textlabel">Reason</td>
        <td colspan="4">
            <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50"
style="width: 100%; height: 60px !important; resize:
vertical"></textarea>
        </td>
    </tr>
</tbody>
</table>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Apply validations on editor template fields

In the following code example, validation has been added to the status field.

INDEX.TS

```

import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { FormValidator } from '@syncfusion/ej2-inputs';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, PopupOpenEventArgs,
EJ2Instance } from '@syncfusion/ej2-schedule';
import { eventData } from '../datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);

```



```

let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    showQuickInfo: false,
    editorTemplate: '#EventEditorTemplate',
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'Editor') {
            if
(!isNullOrUndefined(document.getElementById("EventType_Error"))) {
                document.getElementById("EventType_Error").style.display =
"none";
                document.getElementById("EventType_Error").style.left =
"351px";
            }
            let formElement: HTMLElement =
<HTMLElement>args.element.querySelector('.e-schedule-form');
            let statusElement: HTMLInputElement =
args.element.querySelector('#EventType') as HTMLInputElement;
            if (!statusElement.classList.contains('e-dropdownlist')) {
                let dropDownListObject: DropDownList = new DropDownList({
                    placeholder: 'Choose status', value:
statusElement.value,
                    dataSource: ['New', 'Requested', 'Confirmed'],
                    select: function(args) {
                        if
(!isNullOrUndefined(document.getElementById("EventType_Error"))) {
                            document.getElementById("EventType_Error").style.display = "none";
                        }
                    }
                });
                dropDownListObject.appendTo(statusElement);
            }
            let validator: FormValidator = ((formElement as
EJ2Instance).ej2_instances[0] as FormValidator);
            validator.addRules('EventType', { required: true });
            let startElement: HTMLInputElement =
args.element.querySelector('#StartTime') as HTMLInputElement;
            if (!startElement.classList.contains('e-datetimepicker')) {
                new DateTimePicker({ value: new Date(startElement.value) ||
new Date() }, startElement);
            }
            let endElement: HTMLInputElement =
args.element.querySelector('#EndTime') as HTMLInputElement;
            if (!endElement.classList.contains('e-datetimepicker')) {
                new DateTimePicker({ value: new Date(endElement.value) ||
new Date() }, endElement);
            }
        },
        eventSettings: { dataSource: eventData }
    });
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .custom-event-editor .e-textlabel {
      padding-right: 15px;
      text-align: right;
    }
    .custom-event-editor td {
      padding: 7px;
      padding-right: 16px;
    }
  </style>
  <script id="EventEditorTemplate" type="text/x-template">
    <table class="custom-event-editor" width="100%" cellpadding="5">
      <tbody>
        <tr>
          <td class="e-textlabel">Summary</td>
          <td colspan="4">
            <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
          </td>
        </tr>
        <tr>
          <td class="e-textlabel">Status</td>
          <td colspan="4">
            <input type="text" id="EventType" name="EventType"
class="e-field" style="width: 100%" />
          </td>
        </tr>
        <tr>
          <td class="e-textlabel">From</td>
          <td colspan="4">

```

```

        <input id="StartTime" class="e-field" type="text"
name="StartTime" />
        </td>
    </tr>
    <tr>
        <td class="e-textlabel">To</td>
        <td colspan="4">
            <input id="EndTime" class="e-field" type="text"
name="EndTime" />
        </td>
    </tr>
    <tr>
        <td class="e-textlabel">Reason</td>
        <td colspan="4">
            <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50"
            style="width: 100%; height: 60px !important; resize:
vertical"></textarea>
        </td>
    </tr>
</tbody>
</table>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to save the customized event editor using template

The **e-field** class is not added to each field defined within the template, so you should allow to set those field values externally by using the **popupClose** event.

Note: You can allow to retrieve the data only on the **save** and **delete** option. Data cannot be retrieved on the **close** and **cancel** options in the editor window.

The following code example shows how to save the customized event editor using a template by the **popupClose** event.

INDEX.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
```

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, PopupOpenEventArgs,
PopupCloseEventArgs } from '@syncfusion/ej2-schedule';
import { eventData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    showQuickInfo: false,
    editorTemplate: '#EventEditorTemplate',
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'Editor') {
            let subjectElement: HTMLInputElement =
args.element.querySelector('#Subject') as HTMLInputElement;
            if (subjectElement) {
                subjectElement.value = ((<{ [key: string]: Object
}>(args.data)).Subject as string) || "";
            }
            let statusElement: HTMLInputElement =
args.element.querySelector('#EventType') as HTMLInputElement;
            if (!statusElement.classList.contains('e-dropdownlist')) {
                let dropDownListObject: DropDownList = new DropDownList({
                    placeholder: 'Choose status', value: ((<{ [key: string]:
Object }>(args.data)).EventType as string),
                    dataSource: ['New', 'Requested', 'Confirmed']
                });
                dropDownListObject.appendTo(statusElement);
            }
            let startElement: HTMLInputElement =
args.element.querySelector('#StartTime') as HTMLInputElement;
            if (!startElement.classList.contains('e-datetimepicker')) {
                startElement.value = (<{ [key: string]: Object
}>(args.data)).StartTime as string;
                new DateTimePicker({ value: new Date(startElement.value) ||
new Date(), startElement);
            }
            let endElement: HTMLInputElement =
args.element.querySelector('#EndTime') as HTMLInputElement;
            if (!endElement.classList.contains('e-datetimepicker')) {
                endElement.value = (<{ [key: string]: Object
}>(args.data)).EndTime as string;
                new DateTimePicker({ value: new Date(endElement.value) ||
new Date(), endElement);
            }
            let descriptionElement: HTMLInputElement =
args.element.querySelector('#Description') as HTMLInputElement;
            if (descriptionElement) {
                descriptionElement.value = (<{ [key: string]: Object
}>(args.data)).Description as string || "";
            }
        }
    },
    popupClose: (args: PopupCloseEventArgs) => {
        if (args.type === 'Editor' && !isNullOrUndefined(args.data)) {

```

```

        let subjectElement: HTMLInputElement =
args.element.querySelector('#Subject') as HTMLInputElement;
        if (subjectElement) {
            (<{ [key: string]: Object }>(args.data)).Subject =
subjectElement.value;
        }
        let statusElement: HTMLInputElement =
args.element.querySelector('#EventType') as HTMLInputElement;
        if (statusElement) {
            (<{ [key: string]: Object }>(args.data)).EventType as
string) = statusElement.value;
        }
        let startElement: HTMLInputElement =
args.element.querySelector('#StartTime') as HTMLInputElement;
        if (startElement) {
            (<{ [key: string]: Object }>(args.data)).StartTime =
startElement.value;
        }
        let endElement: HTMLInputElement =
args.element.querySelector('#EndTime') as HTMLInputElement;
        if (endElement) {
            (<{ [key: string]: Object }>(args.data)).EndTime =
endElement.value;
        }
        let descriptionElement: HTMLInputElement =
args.element.querySelector('#Description') as HTMLInputElement;
        if (descriptionElement) {
            (<{ [key: string]: Object }>(args.data)).Description as
string) = descriptionElement.value;
        }
    },
    eventSettings: { dataSource: eventData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .custom-event-editor .e-textlabel {
        padding-right: 15px;
        text-align: right;
    }
    .custom-event-editor td {
        padding: 7px;
        padding-right: 16px;
    }
</style>
<script id="EventEditorTemplate" type="text/x-template">
    <table class="custom-event-editor" width="100%" cellpadding="5">
        <tbody>
            <tr>
                <td class="e-textlabel">Summary</td>
                <td colspan="4">
                    <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">Status</td>
                <td colspan="4">
                    <input type="text" id="EventType" name="EventType"
class="e-field" style="width: 100%" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">From</td>
                <td colspan="4">
                    <input id="StartTime" class="e-field" type="text"
name="StartTime" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">To</td>
                <td colspan="4">
                    <input id="EndTime" class="e-field" type="text"
name="EndTime" />
                </td>
            </tr>
            <tr>
                <td class="e-textlabel">Reason</td>
                <td colspan="4">
                    <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50"
style="width: 100%; height: 60px !important; resize:
vertical"></textarea>
                </td>
            </tr>
        </tbody>
    </table>

```

```

        </tr>
    </tbody>
</table>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the `popupClose` event are as follows.

Type	Description
Editor	For Detailed editor window.
QuickInfo	For Quick popup which opens on cell click.
EditEventInfo	For Quick popup which opens on event click.
ViewEventInfo	For Quick popup which opens on responsive mode.
EventContainer	For more event indicator popup.
RecurrenceAlert	For edit recurrence event alert popup.
DeleteAlert	For delete confirmation popup.
ValidationAlert	For validation alert popup.
RecurrenceValidationAlert	For recurrence validation alert popup.

Quick popups

The quick info popups are the ones that gets opened, when a cell or appointment is single clicked on the desktop mode. On single clicking a cell, you can simply provide a subject and save it. Also, while single clicking on an event, a popup will be displayed where you can get the overview of the event information. You can also edit or delete those events through the options available in it.

By default, these popups are displayed over cells and appointments of Scheduler and to disable this action, set `false` to `showQuickInfo` property.

The quick popup that opens while single clicking on the cells are not applicable on mobile devices.

INDEX.TS

```
import { Schedule, Day, Week, TimelineViews, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    showQuickInfo: false,
    views: ['TimelineDay', 'Day', 'Week', 'Month', 'Agenda'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .e-textlabel {
            font-weight: bold;
            padding-right: 5px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```



```
<body>

  <div id="container">
    <div id="button"></div>
    <div id="Schedule"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Open quick popup on multiple cell selection

You can display the immediate quick popup after multiple cells selected in scheduler, by setting the `quickInfoOnSelectionEnd` property to true. By default, it's value set to false.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 3, 1),
  quickInfoOnSelectionEnd : true,
  quickInfoTemplates: {
    header: '#headerTemplate',
    content: '#contentTemplate',
    footer: '#footerTemplate'
  },
  group: {
    resources: ['Rooms', 'Owners']
  },
  resources: [
    {
      field: 'RoomId', title: 'Room',
      name: 'Rooms', allowMultiple: false,
      dataSource: [
        { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
        { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
      ],
      textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
    }, {
      field: 'OwnerId', title: 'Owner',
      name: 'Owners', allowMultiple: true,
      dataSource: [
        { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
        '#ffaa00' },
        { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
        '#f8a398' },
        { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
        OwnerColor: '#7499e1' }
      ]
    }
  ],
```

```

        textField: 'OwnerText', idField: 'Id', groupIDField:
'OwnerGroupId', colorField: 'OwnerColor'
    }
    ],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .e-textlabel {
            font-weight: bold;
            padding-right: 5px;
        }
        .custom-event-editor td {
            padding: 5px;
        }
        .e-quick-popup-wrapper .e-cell-content {
            padding: 0 10px 10px 10px;
        }
        .e-quick-popup-wrapper .e-cell-content div {
            padding-bottom: 10px;
        }
        .e-quick-popup-wrapper .e-cell-content .e-field {
            border-left-width: 0;
            border-top-width: 0;
            border-right-width: 0;
            height: 25px;
            width: 100%;
        }
    </style>

```

```

.e-quick-popup-wrapper .e-event-content {
    display: flex;
}
.e-quick-popup-wrapper .e-event-header {
    position: absolute;
    right: 0;
}
.e-quick-popup-wrapper .e-cell-footer .e-event-details {
    padding-right: 40px;
}
.e-quick-popup-wrapper .e-cell-footer .e-event-create,
.e-quick-popup-wrapper .e-event-footer .e-event-edit {
    position: absolute;
    right: 0;
}
.e-quick-popup-wrapper .e-event-content .subject {
    padding-top: 10px;
    padding-left: 10px;
    font-weight: 500;
    font-size: 14px;
}
</style>
<script id="headerTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-header">
            <div class="e-header-icon-wrapper">
                <button class="e-close" title="Close"></button>
            </div>
        </div>
        ${else}
        <div class="e-event-header">
            <div class="e-header-icon-wrapper">
                <button class="e-close" title="CLOSE"></button>
            </div>
        </div>
        ${/if}
    </div>
</script>
<script id="contentTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-content">
            <form class="e-schedule-form">
                <div>
                    <input class="subject e-field" type="text"
name="Subject" placeholder="Title">
                </div>
                <div>
                    <input class="location e-field" type="text"
name="Location" placeholder="Location">
                </div>
            </form>
        </div>
        ${else}
        <div class="e-event-content">
            <div class="e-subject-wrap">

```

```

        ${if (Subject)}
        <div class="subject">${Subject}</div>${/if} ${if (City)}
        <div class="location">${City}</div>${/if} ${if
(Description)}
        <div class="description">${Description}</div>${/if}
        </div>
    </div>
    </script>
    <script id="footerTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-footer">
            <button class="e-event-details" title="Additional
Details">Additional Details</button>
            <button class="e-event-create" title="Add">Add</button>
        </div>
        ${else}
        <div class="e-event-footer">
            <button class="e-event-edit" title="Edit">Edit</button>
            <button class="e-event-delete"
title="Delete">Delete</button>
        </div>
        ${/if}
    </div>
    </script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

How to change the watermark text of quick popup subject

By default, Add Title text is displayed on the subject field of quick popup. To change the default watermark text, change the value of the appropriate localized word collection used in the Scheduler.

```
`ts
```

```
L10n.load({
```

```
'en-US': {
```

```
'schedule': {
  'addTitle' : 'New Title'
}
}
});
`
```

Customizing quick popups

The look and feel of the built-in quick popup window, which opens when single clicked on the cells or appointments can be customized by making use of the `quickInfoTemplates` property of the Scheduler. There are 3 sub-options available to customize them easily,

- header - Accepts the template design that customizes the header part of the quick popup.
- content - Accepts the template design that customizes the content part of the quick popup.
- footer - Accepts the template design that customizes the footer part of the quick popup.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 3, 1),
  quickInfoTemplates: {
    header: '#headerTemplate',
    content: '#contentTemplate',
    footer: '#footerTemplate'
  },
  group: {
    resources: ['Rooms', 'Owners']
  },
  resources: [
    {
      field: 'RoomId', title: 'Room',
      name: 'Rooms', allowMultiple: false,
      dataSource: [
        { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
        { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
      ],
      textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
    }, {
      field: 'OwnerId', title: 'Owner',
      name: 'Owners', allowMultiple: true,
      dataSource: [
        { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
        '#ffaa00' },
        { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
        '#f8a398' },
        { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1,
        OwnerColor: '#7499e1' }
      ]
    }
  ]
});
```

```

        textField: 'OwnerText', idField: 'Id', groupIDField:
        'OwnerGroupId', colorField: 'OwnerColor'
    }
    ],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .e-textlabel {
            font-weight: bold;
            padding-right: 5px;
        }
        .custom-event-editor td {
            padding: 5px;
        }
        .e-quick-popup-wrapper .e-cell-content {
            padding: 0 10px 10px 10px;
        }
        .e-quick-popup-wrapper .e-cell-content div {
            padding-bottom: 10px;
        }
        .e-quick-popup-wrapper .e-cell-content .e-field {
            border-left-width: 0;
            border-top-width: 0;
            border-right-width: 0;
            height: 25px;
            width: 100%;
        }
    </style>

```

```

.e-quick-popup-wrapper .e-event-content {
    display: flex;
}
.e-quick-popup-wrapper .e-event-header {
    position: absolute;
    right: 0;
}
.e-quick-popup-wrapper .e-cell-footer .e-event-details {
    padding-right: 40px;
}
.e-quick-popup-wrapper .e-cell-footer .e-event-create,
.e-quick-popup-wrapper .e-event-footer .e-event-edit {
    position: absolute;
    right: 0;
}
.e-quick-popup-wrapper .e-event-content .subject {
    padding-top: 10px;
    padding-left: 10px;
    font-weight: 500;
    font-size: 14px;
}
</style>
<script id="headerTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-header">
            <div class="e-header-icon-wrapper">
                <button class="e-close" title="Close"></button>
            </div>
        </div>
        ${else}
        <div class="e-event-header">
            <div class="e-header-icon-wrapper">
                <button class="e-close" title="CLOSE"></button>
            </div>
        </div>
        ${/if}
    </div>
</script>
<script id="contentTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-content">
            <form class="e-schedule-form">
                <div>
                    <input class="subject e-field" type="text"
name="Subject" placeholder="Title">
                </div>
                <div>
                    <input class="location e-field" type="text"
name="Location" placeholder="Location">
                </div>
            </form>
        </div>
        ${else}
        <div class="e-event-content">
            <div class="e-subject-wrap">

```

```

        ${if (Subject)}
        <div class="subject">${Subject}</div>${/if} ${if (City)}
        <div class="location">${City}</div>${/if} ${if
(Description)}
        <div class="description">${Description}</div>${/if}
        </div>
    </div>
    </script>
    <script id="footerTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-footer">
            <button class="e-event-details" title="Additional
Details">Additional Details</button>
            <button class="e-event-create" title="Add">Add</button>
        </div>
        ${else}
        <div class="e-event-footer">
            <button class="e-event-edit" title="Edit">Edit</button>
            <button class="e-event-delete"
title="Delete">Delete</button>
        </div>
        ${/if}
    </div>
    </script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

The quick popup in adaptive mode can also be customized using `quickInfoTemplates` using `e-device` class.

More events indicator and popup

When the number of appointments count that lies on a particular time range * default appointment height exceeds the default height of a cell in month view and all other timeline views, a + more text indicator will be displayed at the bottom of those cells. This indicator denotes that the cell contains few

more appointments in it and clicking on that will display a popup displaying all the appointments present on that day.

To disable this option of showing popup with all hidden appointments, while clicking on the text indicator, you can do code customization within the `popupOpen` event.

The same indicator is displayed on all-day row in calendar views such as day, week and work week views alone, when the number of appointment count present in a cell exceeds three. Clicking on the text indicator here will not open a popup, but will allow the expand/collapse option for viewing the remaining appointments present in the all-day row.

The following code example shows how to disable the display of such popups while clicking on the more text indicator.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, PopupOpenEventArgs } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'Month',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'EventContainer') {
            args.cancel = true;
        }
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-textlabel {
        font-weight: bold;
        padding-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to customize the popup that opens on more indicator

The following code example shows you how to customize the default more indicator popup in which number of events rendered count on the day has been shown in the header.

INDEX.TS

```

import { Internationalization } from '@syncfusion/ej2-base';
import { Schedule, Month, PopupOpenEventArgs } from '@syncfusion/ej2-
schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject( Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Month'],
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'EventContainer') {
            let instance: Internationalization = new Internationalization();
            let date: string = instance.formatDate(args.data.date, {
                skeleton: 'MMEd' });
            ((args.element.querySelector('.e-header-date')) as
            HTMLElement).innerText = date;
        }
    }
});

```

```

        ((args.element.querySelector('.e-header-day')) as
HTMLInputElement).innerText = 'Event count: ' + args.data.event.length;
    }
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .e-textlabel {
            font-weight: bold;
            padding-right: 5px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to close the popup manually

The following code example demonstrates the how to close popup manually.

INDEX.TS

```

import { Schedule, Month, PopupCloseEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject( Month);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Month'],
  popupClose: (args: PopupCloseEventArgs) => {
    console.log(arg);
  },
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-textlabel {

```

```

        font-weight: bold;
        padding-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to display the popup when clicking on the more text indicator

The following code example demonstrates the open popup when clicking the more text indicator. By default, scheduler set to open the popup on clicking the more text indicator.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, MoreEventsClickArgs } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'Month',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-textlabel {
        font-weight: bold;
        padding-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to prevent the display of popup when clicking on the more text indicator

It is possible to prevent the display of popup window by passing the value `true` to `cancel` option within the `MoreEventsClick` event.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, MoreEventsClickArgs } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({

```

```

width: '100%',
height: '550px',
selectedDate: new Date(2018, 1, 15),
currentView: 'Month',
views: ['Day', 'Week', 'WorkWeek', 'Month'],
moreEventsClick: (args: MoreEventsClickArgs) => {
    args.cancel = true;
},
eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-textlabel {
      font-weight: bold;
      padding-right: 5px;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="button"></div>
    <div id="Schedule"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to navigate Day view when clicking on more text indicator

The following code example shows you how to customize the `moreEventsClick` property to navigate to the Day view when clicking on the more text indicator.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, MoreEventsClickArgs } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'Month',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    moreEventsClick: (args: MoreEventsClickArgs) => {
        args.isPopupOpen = false;
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-textlabel {
        font-weight: bold;
        padding-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to close the editor window manually

You can close the editor window by using `closeEditor()` method.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, MoreEventsClickArgs } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'Month',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: { dataSource: scheduleData },
    popupOpen: (args: PopupOpenEventArgs) => {
        if (args.type === 'Editor') {
            if (!args.element.querySelector('#closeEditor')) {
                let btnElement = createElement("BUTTON", { id: 'closeEditor',
                    className: 'e-custom-close' });
            }
        }
    }
});

```

```

        args.element.querySelector('.e-footer-
content').appendChild(btnElement)
        let btnObj: Button = new Button();
        btnElement.textContent = "Close Editor";
        btnObj.appendTo('#closeEditor');
        btnObj.element.onclick = (): void => {
            scheduleObj.closeEditor();
        }
    }
},
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .e-textlabel {
            font-weight: bold;
            padding-right: 5px;
        }
    </style>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to open the quick info popup manually

You can open the quick info popup in scheduler by using the [openQuickInfoPopup](#) public method. To open the cell quick info popup, you can pass the cell data as an argument to the method. To open the event quick info popup, you should pass the event data object as an argument to the method.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { createElement } from '@syncfusion/ej2-base';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2023, 2, 5),
    currentView: 'Month',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: {
        dataSource: [{
            Id: 1,
            Subject: 'Review Meeting',
            StartTime: new Date(2023, 2, 5, 20, 0, 0),
            EndTime: new Date(2023, 2, 5, 21, 0, 0)
        }]
    }
});
scheduleObj.appendTo('#Schedule');
let btnElement = createElement("Button", { id: 'openCellClickInfo' });
document.getElementById('button').appendChild(btnElement);
btnElement.textContent = "Show Cell QuickInfo Popup";
btnElement.onclick = () => {
    scheduleObj.openQuickInfoPopup({
        Subject: 'Meeting',
        StartTime: new Date(2023, 2, 6, 20, 0, 0),
        EndTime: new Date(2023, 2, 6, 21, 0, 0)
    });
};
let eventButton = createElement("Button", { id: 'openeventClickInfo' });
document.getElementById('button').appendChild(eventButton);
eventButton.textContent = "Show Event QuickInfo Popup";
eventButton.onclick = () => {
    scheduleObj.openQuickInfoPopup({
        Id: 1,
        Subject: 'Review Meeting',

```

```

        StartTime: new Date(2023, 2, 5, 20, 0, 0),
        EndTime: new Date(2023, 2, 5, 21, 0, 0)
    });
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .e-textlabel {
            font-weight: bold;
            padding-right: 5px;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

How to close the quick info popup manually

You can close the quick info popup in scheduler by using the `closeQuickInfoPopup()` public method. The following code example demonstrates the how to close quick info popup manually.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, MoreEventsClickArgs } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
import { createElement } from '@syncfusion/ej2-base';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  currentView: 'Month',
  views: ['Day', 'Week', 'WorkWeek', 'Month'],
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let closeButton = createElement("Button", { id: 'closeQuickInfo' });
document.getElementById('button').appendChild(closeButton);
closeButton.textContent = "Close Quick Info Popup";
closeButton.onclick = (): void => {
  scheduleObj.closeQuickInfoPopup();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
```

```

<style>
    .e-textlabel {
        font-weight: bold;
        padding-right: 5px;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="button"></div>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Timezone in EJ2 JavaScript Schedule control

The Scheduler makes use of the current system time zone by default. If it needs to follow some other user-specific time zone, then the `timezone` property needs to be used. Apart from the default action of applying specific timezone to the Scheduler, it is also possible to set different time zone values for each appointments through the properties `startTimezone` and `endTimezone` which can be defined as separate fields within the event fields collection.

Note: `timezone` property only applicable for the appointment processing and current time indication.

Understanding date manipulation in JavaScript

The `new Date()` in JavaScript returns the exact current date object with complete time and timezone information. For example, it may return value such as `Wed Dec 12 2018 05:23:27 GMT+0530 (India Standard Time)` which indicates that the current date is December 12, 2018 and the current time is 5.23 AM on browsers following the IST timezone.

Scheduler with no timezone

When no specific time zone is set to Scheduler, appointments will be displayed based on the client system's timezone which is the default behavior. Here, the same appointment when viewed from different timezone will have different start and end times.

The following code example displays an appointment from 9.00 AM to 10.00 AM when you open the Scheduler from any of the timezone. This is because, we are providing the start and end time enclosing with `new Date()` which works based on the client browser's timezone.

INDEX.TS

```
import { Schedule, Day, Week, TimelineViews, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 9, 0),
    EndTime: new Date(2018, 1, 15, 10, 0)
}];
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
    eventSettings: {
        dataSource: data
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scheduler set to specific timezone

When a time zone is set to Scheduler through `timezone` property, the appointments will be displayed exactly based on the Scheduler timezone regardless of its client timezone. In the following code example, appointments will be displayed based on Eastern Time (UTC -05:00).

INDEX.TS

```

import { Schedule, Day, Week, Month, Timezone } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Day', 'Week', 'Month'],
    selectedDate: new Date(2018, 1, 17),
    timezone: 'America/New_York',
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display events on same time everywhere with no time difference

Setting **timezone** to UTC for Scheduler will display the appointments on same time as in the database for all the users in different time zone.

INDEX.TS

```

import { Schedule, Day, Week, Month, Timezone } from '@syncfusion/ej2-
schedule';
import { fifaEventsData } from './datasource.ts';
import { extend } from '@syncfusion/ej2-base';
Schedule.Inject(Day, Week, Month);
let fifaEvents: Object[] = <Object[]>extend([], fifaEventsData, null, true);
let timezone: Timezone = new Timezone();
for (let fifaEvent of fifaEvents) {
    let event: { [key: string]: Object } = fifaEvent as { [key: string]:
Object };
    event.StartTime = timezone.removeLocalOffset(<Date>event.StartTime);
    event.EndTime = timezone.removeLocalOffset(<Date>event.EndTime);
}
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 5, 17),
    views: ['Day', 'Week', 'Month'],
    timezone: 'UTC',
    eventSettings: { dataSource: fifaEvents }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Set specific timezone for events

It is possible to set different timezone for Scheduler events by setting `startTimezone` and `endTimezone` properties within the `eventSettings` option. It allows each appointment to maintain different timezone and displays on Scheduler with appropriate time differences.

INDEX.TS

```
import { Schedule, Day, Week, TimelineViews, Month, Agenda } from
 '@syncfusion/ej2-schedule';
let data: object [] = [{
    Subject: 'Paris',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 12, 30),
    StartTimezone: 'Europe/Moscow',
    EndTimezone: 'Europe/Moscow'
}];
Schedule.Inject(Day, Week, TimelineViews, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'TimelineWeek', 'Month', 'Agenda'],
    eventSettings: {
        dataSource: data
    }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add or remove timezone names to/from the timezone collection

Instead of displaying all the timezone names within the timezone collection (more than 200 are displayed on the editor window timezone fields by default), you can customize the timezone collection at application end as shown in the following example.

INDEX.TS

```

import { Schedule, Day, Week, Month, timezoneData } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
let customTimezoneData: { [key: string]: Object }[] = [
    { Value: 'America/New_York', Text: '(UTC-05:00) Eastern Time' },
    { Value: 'UTC', Text: 'UTC' },
    { Value: 'Asia/Kolkata', Text: '(UTC+05:30) India Standard Time' }
];
timezoneData.splice(0, timezoneData.length, ...customTimezoneData);
Schedule.Inject(Day, Week, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 11),
    views: ['Day', 'Week', 'Month'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timezone methods

offset

This method is used to calculate the difference between passed UTC date and timezone.

| Parameters | Type | Description |

|-----|-----|-----|

| Date | Date | UTC time as date object. |

| Timezone | String | Timezone. |

Returns **number**

```
`ts
```

```
// Assume your local timezone as IST/UTC+05:30
```

```
let timezone: Timezone = new Timezone();
```

```
let date: Date = new Date(2018,11,5,15,25,11);
```

```
let timeZoneOffset: number = timezone.offset(date,"Europe/Paris");
```

```
console.log(timeZoneOffset); //-60
```

```
,
```

convert

This method is used to convert the passed date from one timezone to another timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
fromOffset	number/string	Timezone from which date need to be converted.
toOffset	number/string	Timezone to which date need to be converted.

Returns **Date**

```
`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let convertedDate: Date = timezone.convert(date, "Europe/Paris", "Asia/Tokya");
let convertedDate1: Date = timezone.convert(date, 60, -360);
console.log(convertedDate); //2018-12-05T08:55:11.000Z
console.log(convertedDate1); //2018-12-05T16:55:11.000Z
`
```

add

This method is used to add the time difference between passed UTC date and timezone.

Parameters	Type	Description
----- ----- -----		
Date	Date	UTC time as date object.
Timezone	String	Timezone.

Returns **Date**

```
`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let convertedDate: Date = timezone.add(date, "Europe/Paris");
console.log(convertedDate); //2018-12-05T05:25:11.000Z
`
```

remove

This method is used to remove the time difference between passed UTC date and timezone.

Parameters	Type	Description
------------	------	-------------

----- ----- -----	----- ----- -----	----- ----- -----
Date	Date	UTC as date object.
Timezone	String	Timezone.

Returns **Date**

```
`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let convertedDate: Date = timezone.remove(date, "Europe/Paris");
console.log(convertedDate); //2018-12-05T14:25:11.000Z
`
```

removeLocalOffset

This method is used to remove the local offset time from the date passed.

----- ----- -----	----- ----- -----	----- ----- -----
Date	Date	UTC as date object.

Returns **Date**

```
`ts
// Assume your local timezone as IST/UTC+05:30
let timezone: Timezone = new Timezone();
let date: Date = new Date(2018,11,5,15,25,11);
let convertedDate: Date = timezone.removeLocalOffset(date);
console.log(convertedDate); //2018-12-05T15:25:11.000Z
`
```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Views in EJ2 JavaScript Schedule control

The Scheduler includes wide variety of view modes with unique configuration options for each view. The available view modes are Day, Week, Work Week, Month, Agenda, Month Agenda, Timeline Day, Timeline Week, Timeline Work Week and Timeline Month, out of which the **Week** view is set as active.

To navigate between different views and dates, the navigation options are available at the Scheduler header bar. The active view option is usually highlighted by default. The date range of the active view will also be displayed at the left corner of the header bar, clicking on which will open a calendar popup for the ease of desired date selection.

By default, Scheduler displays the calendar views such as day, week, work week, month and agenda.

Setting specific view on scheduler

As the Scheduler displays **week** view by default, therefore to change the active view, set **currentView** property with the desired view name. The applicable view names that the Scheduler accepts are as follows,

- Day
- Week
- WorkWeek
- Month
- Year
- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear

It is necessary to import and inject the appropriate view modules into the application to make use of these view modes on the Scheduler. Also, it is possible to display only the desired views on the Scheduler. To define and configure specific views, use the [views](#) property.

In the following example, the Scheduler displays 4 views namely, Week, Month, TimelineWeek and TimelineMonth. The appropriate view modules are imported and injected properly to display those views on the Scheduler.

INDEX.TS

```
import { Schedule, Week, Month, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject( Week, Month, TimelineViews, TimelineMonth );
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Week', 'TimelineWeek', 'Month', 'TimelineMonth'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To configure Scheduler with simply 2 views, but with different configurations on each view, refer the following code example. Here, the Week view displays the dates in **dd-MM-yyyy** format whereas the Month view hides the weekend days and also displays it in readonly mode.

INDEX.TS

```

import { Schedule, Week, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject( Week, Month );
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'Week', dateFormat: 'dd-MMM-yyyy' }, { option:
'Month', showWeekend: false, readonly: true }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

View specific configuration

There are scenarios where each view may need to have different configurations. For such cases, you can define the applicable scheduler properties within the **views** Property for each view option as depicted in the following examples. The fields available to be used within each view options are as follows.

Property	Type	Description	Applicable views
-----	-----	-----	-----

- | **option** | View | It accepts the Scheduler view name, based on which we can define its related properties. The view names can be **Day**, **Week** and so on. | All views. |
- | **isSelected** | Boolean | It acts similar to the **currentView** property and defines the active view of the Scheduler. | All views. |
- | **dateFormat** | Date | By default, Scheduler follows the date format as per the default culture assigned to it. When it is defined under specific view, only those assigned views follows this date format. | All views. |
- | **readonly** | Boolean | When set to **true**, prevents the CRUD actions on the respective view under where it is defined. | All views. |
- | **resourceHeaderTemplate** | String | The template option which is used to customize the resource header cells on the Scheduler. It gets applied only on the views, wherever it is defined. | All views. |
- | **dateHeaderTemplate** | String | The template option which is used to customize the date header cells and is applied only on the views, wherever it is defined. | All views. |
- | **eventTemplate** | String | The template option to customize the events background. It will get applied to the events of the view to which it is currently being defined. | All views. |
- | **showWeekend** | Boolean | When set to **false**, it hides the weekend days of a week from the views on which it is defined. | All views. |
- | **group** | [GroupModel](#) | Allows to set different resource grouping options on all available Scheduler view modes. | All views. |
- | **cellTemplate** | String | The template option to customize the work cells of the Scheduler and is applied only on the views, on which it is defined. | Applicable on all views except Agenda view. |
- | **workDays** | Number[] | It is used to set the working days on the Scheduler views. | Applicable on all views except Agenda view. |
- | **displayName** | String | When a particular view is customized to display with different intervals, this property allows the user to set different display name for each of the views. | Applicable on all views except Agenda and Month Agenda. |
- | **interval** | Number | It allows to customize the default Scheduler views with different set of days, weeks, work weeks or months on the applicable view type. | Applicable on all views except Agenda and Month Agenda. |
- | **startHour** | String | It is used to specify the start hour, from which the Scheduler should be displayed. It accepts the time string in a short skeleton format and also, hides the time beyond the specified start time. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. |
- | **endHour** | String | It is used to specify the end hour, at which the Scheduler ends. It accepts the time string in a short skeleton format. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |
- | **timeScale** | [TimeScaleModel](#) | Allows to set different timescale configuration on each applicable view modes. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |

| **showWeekNumber** | Boolean | When set to **true**, shows the week number on the respective weeks. | Applicable on Day, Week, Work Week, and Month views. |

| **allowVirtualScrolling** | Boolean | It is used to enable or disable the virtual scrolling functionality. | Applicable on Agenda and Timeline views. |

| **headerRows** | [HeaderRowsModel](#) | Allows defining the custom header rows on timeline views of the Scheduler to display the year, month, week, date and hour label as an individual row. | Applicable only on all timeline views. |

Day view

Usually a day view displays a single day with all its related appointments. It is possible to customize the day view to display more number of days by extending the **views** property with **interval** option. You can also define any of the above defined properties within the **views** object definition as depicted in the following code example.

INDEX.TS

```
import { Schedule, Day } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    currentView: 'Day',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'Day', startHour: '09:30', endHour: '18:00',
timeScale: { enable: true, slotCount: 5 } }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

All the above defined properties can be accessed within Day view except `allowVirtualScrolling` and `headerRows`.

Week view

The Week view displays a count of 7 days (from Sunday to Saturday) with all its related appointments. The first day of the week can be changed using the `firstDayOfWeek` which accepts the integer (Sunday=0, Monday=1, Tuesday=2 and so on) value. You can navigate to a particular date in day view from the week view by clicking on the appropriate dates on the date header bar.

INDEX.TS

```

import { Schedule, Day, Week } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [
        { option: 'Day', interval: 2, displayName: '2 Days', startHour:
'09:30', endHour: '18:00', timeScale: {enable: true, slotCount: 5}},
        { option: 'Week', displayName: '2 Weeks', interval: 2, showWeekend:
false, isSelected: true }
    ],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

All the above defined properties in the table can be accessed within Week and Work week views except `allowVirtualScrolling` and `headerRows`.

Work Week view

The Work week view displays only the working days of a week (count of 5 days) and its associated appointments. It is possible to customize the working days on the work week view by using the `workDays` property which accepts an array of integer values (such as Sunday=0, Monday=1, Tuesday=2

and so on). By default, it displays from Monday to Friday (5 days). You can also navigate to a particular date in the day view from the work week view by clicking on the appropriate dates in the date header bar.

The following code example depicts how to change the working days only on the **Work Week** view of the Scheduler.

INDEX.TS

```
import { Schedule, WorkWeek } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'WorkWeek', workDays: [2, 3, 5] }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

        <div id="container">
            <div id="Schedule"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The Week, Work week and Day views can display the all-day row appointments in a separate all-day row with an expand/collapse option to view it.

Month view

A Month view displays the entire days of a particular month and all its related appointments. You can navigate to a particular date in the day view by clicking on the appropriate date text on the month cells.

By default, when you try to create an appointment through Month view, it is considered as created for an entire day. You can explicitly change this behavior by unchecking the **All-day** option from editor window, so that it defaults to the start time duration as 9.00 AM and end time as 9.30 AM.

You can also have the **+ more** text indicator on each day cell of a Month view, clicking on which will allow you to view the hidden appointments of a day.

INDEX.TS

```

import { Schedule, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'Month', showWeekNumber: true, readonly: true }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Year view

A Year view displays all the days of a particular year with months and all its related appointments. You can navigate to a particular date in the day view by clicking on the appropriate date text on the year cells.

Year view is available in both the **Horizontal** and **Vertical** orientations. You can manage the orientation of year view through **views** property.

INDEX.TS

```

import { Schedule, Year } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Year);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Year'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The year view also has module support. In that, you can get all the months of a particular year in a calendar view format. In that calendar view, appointment contained dates are highlighted with dots placed under the individual date. When you click on the date, the event popup will be displayed and the events will be listed.

Agenda view

The Agenda view lists out the appointments in a grid-like view for the next 7 days by default from the current date. The count of the days can be changed using the API `agendaDaysCount`. It allows virtual scrolling of dates by enabling the `allowVirtualScrolling` property. Also, you can enable or disable the display of days on Scheduler that has no appointments by setting true or false to the `hideEmptyAgendaDays` property.

The following code example depicts how to customize the display of events within Agenda view alone.

INDEX.TS

```
import { Internationalization } from '@syncfusion/ej2-base';
import { Schedule, Agenda } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Agenda);
let instance: Internationalization = new Internationalization();
(window as TemplateFunction).getTimeString = (value: Date) => {
    return instance.formatDate(value, { skeleton: 'hm' });
};
interface TemplateFunction extends Window {
    getTimeString?: Function;
}
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '350px',
    selectedDate: new Date(2018, 1, 15),
    views: [{
        option: 'Agenda',
        eventTemplate: '<div class="template-wrap"><div
class="subject">${Subject}</div><div
class="time">${getTimeString(data.StartTime)} -
${getTimeString(data.EndTime)}</div></div>',
        allowVirtualScrolling: true
    }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Schedule Height is mandatory to set in pixels for Agenda view alone.

Month Agenda view

A Month-Agenda view shows a month calendar, where clicking on a particular day will display the appointments present on that date below the calendar. The day with appointments are differentiated with a circular dot below the date of the calendar.

The following code example shows how to hide the weekend days on **MonthAgenda** view as well as the working days list is modified on Month Agenda view alone.

INDEX.TS

```

import { Schedule, MonthAgenda } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(MonthAgenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 14),
    views: [{ option: 'MonthAgenda', showWeekend: false, workDays: [1, 2, 3]
    }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline views – Day, Week, Work Week

Similar to the day view, timeline day view shows a single day with all its appointments where the time slots are displayed horizontally. By default, the cell height adjusts as per the height set to Scheduler.

When the number of appointments exceeds the visible area of the cells, the **+ more** text indicator will be displayed at the bottom to denote the presence of few more appointments in that time range.

To make use of the timeline views (Timeline Day, Timeline Week and Timeline Work Week) on Scheduler, import and inject the module `TimelineViews` from the `ej2-schedule` package.

INDEX.TS

```
import { Schedule, TimelineViews } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'TimelineDay', startHour: '10:00', endHour: '15:30'
}],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Similar to the Week view, the timeline week view shows 7 days with its associated appointments with the time slots displayed horizontally.

INDEX.TS

```

import { Schedule, TimelineViews } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'TimelineWeek', timeScale: { enable: false } }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following code example depicts how to display the timeline work week view on Scheduler,

INDEX.TS

```

import { Schedule, TimelineViews } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 14),
    views: [{ option: 'TimelineWorkWeek', interval: 3, workDays: [1, 3, 5],
    dateFormat: 'dd-MMM-yyyy' }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clicking on the dates in the date header bar of Timeline day, Timeline week and Timeline work week will allow you to navigate to the Agenda view.

Timeline Month view

A Timeline Month view displays the current month days along with its appointments. To make use of the timeline Month view on Scheduler, import and inject `TimelineMonth` module from the `ej2-schedule` package.

INDEX.TS

```

import { Schedule, TimelineMonth } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'TimelineMonth', showWeekend: false }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Schedule Typescript Component</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Clicking on the dates in the date header bar of Timeline month will allow you to navigate to the Timeline day view.

Timeline Year view

In Timeline Year view, each row depicts a single resource. Whereas in the vertical view, each resource is grouped parallelly as columns. Here, the resource grouping follows the tree-view like hierarchical grouping structure and can contain any level of child resources.

To make use of the timeline Year view on Scheduler, import and inject `TimelineYear` module from the `ej2-schedule` package.

INDEX.TS

```
import { Schedule, TimelineYear } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineYear);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'TimelineYear', displayName: 'Horizontal Timeline
Year', isSelected: true }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline Year view is available in both the **Horizontal** and **Vertical** orientations. You can manage the orientation of Timeline Year view through **views** property.

Resource grouping

The following code example depicts how to group the multiple resources on Timeline Year view and its relevant events are displayed accordingly under those resources.

INDEX.TS

```

import { Schedule, TimelineYear } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(TimelineYear);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: [{ option: 'TimelineYear', displayName: 'Horizontal Timeline Year', isSelected: true },
        { option: 'TimelineYear', displayName: 'Vertical Timeline Year',
orientation: 'Vertical' }],
    selectedDate: new Date(2018, 3, 4),
    group: {
        resources: ['Rooms', 'Owners']
    },
    resources: [{
        field: 'RoomId', title: 'Room',
        name: 'Rooms', allowMultiple: false,
        dataSource: [
            { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
            { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
        ],
        textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
    }, {
        field: 'OwnerId', title: 'Owner',
        name: 'Owners', allowMultiple: true,
        dataSource: [
            { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor: '#ffaa00' },
            { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor: '#f8a398' },
            { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor: '#7499e1' }
        ],
        textField: 'OwnerText', idField: 'Id', groupIDField: 'OwnerGroupId',
        colorField: 'OwnerColor'
    }],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Auto row height

Timeline Year view supports Auto row height. When the feature `rowAutoHeight` is enabled, the row height gets auto-adjusted based on the number of overlapping events occupied in the same time range. If you disable the Auto row height, you have the + more text indicator on each day cell of a Timeline Year view, clicking on which will allow you to view the hidden appointments of a day.

INDEX.TS

```
import { Schedule, TimelineYear } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineYear);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    rowAutoHeight: true,
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'TimelineYear', displayName: 'Horizontal Timeline Year', isSelected: true },
        { option: 'TimelineYear', displayName: 'Vertical Timeline Year',
            orientation: 'Vertical' }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Extending view intervals

It is possible to customize the display of default number of days on different Scheduler view modes. For example, a day view can be extended to display 3 days by setting the `interval` option as 3 for the `Day` option within the `views` property as depicted in the following code example. In the same way, you can also display 2 weeks by setting interval 2 for the `Week` option.

You can provide the alternative display name for such customized views on the Scheduler header bar, by setting the appropriate `displayName` property.

INDEX.TS

```

import { Schedule, Day, Week } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    currentView: 'Day',
    selectedDate: new Date(2018, 1, 15),
    views: [{ option: 'Day', interval: 3, displayName: '3 Days' },
            { option: 'Week', interval: 2, displayName: '2 Weeks' }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The view intervals can be extended on all the Scheduler view modes except Agenda and Month-Agenda views.

We can also use `isSelected` property to set the current view of the scheduler. The below code sample demonstrates that how to use `isSelected` property.

INDEX.TS

```

import { Schedule, Day, Week } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: [
        {
            option: 'Day',
            interval: 3,
            displayName: '3 Days',
        },
        {
            option: 'Week',
            displayName: '2 Weeks',
            interval: 2,
            isSelected: true,
        },
    ],
    eventSettings: { dataSource: scheduleData }
});

```



```
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

See Also

- [How to restrict view navigation while clicking on dates](#)

Calendar mode in EJ2 JavaScript Schedule control

The Scheduler supports the following two types of calendar mode.

- Gregorian Calendar
- Islamic Calendar

Gregorian Calendar

The Scheduler by default displays the gregorian calendar dates which is the most widely used calendar in the world. The Gregorian calendar is a solar calendar which has 12 months with 28 to 31 days each. A year which is divisible by four is said to be a leap year in this calendar mode. A leap year usually has 366 days whereas the regular year has 365 days.

Islamic Calendar

The Islamic Calendar, also known as the Hijri or Muslim calendar, is a lunar calendar which has 12 months in a year with 354 or 355 days. Each month of this calendar denotes the birth of the new lunar cycle and therefore, each month can have 29 or 30 days depending on the visibility of the moon. Here, the odd-numbered months have 30 days and the even months have 29 days.

The current Islamic year is 1440 AH. Usually the Gregorian calendar runs from approximately 11 September 2018 to 30 August 2019 for this 1440 AH year.

The Scheduler has a property `calendarMode` which is used to switch between the gregorian and islamic calendar modes. By default, it is set to `Gregorian` and to use it with Islamic calendar dates, define the `calendarMode` of Scheduler to `Islamic`. The following example depicts, how to display the Islamic calendar dates on Scheduler.

To make use of Islamic calendar in Scheduler, import the `Calendar` and `Islamic` module from `ej2-calendars` package and also inject it using the `Calendar.Inject` method. Apart from this, it requires the following CLDR data to be loaded using `loadCldr` function.

- `numberingSystems.json`
- `ca-gregorian.json`
- `numbers.json`
- `timeZoneNames.json`
- `ca-islamic.json`

To know more information on, how to install the CLDR data, refer the [Internationalization](#) topic.

INDEX.TS

```
import {
    Schedule, Day, Week, WorkWeek, Month, Agenda, TimelineViews,
    TimelineMonth,
    Resize, DragAndDrop, MonthAgenda, CalendarType
} from '@syncfusion/ej2-schedule';
import { Ajax, loadCldr, setCulture, L10n } from '@syncfusion/ej2-base';
import { Calendar, Islamic } from '@syncfusion/ej2-calendars';
```

```

import { scheduleData } from './datasource.ts';
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
import * as islamic from './ca-islamic.json';
Schedule.Inject(Day, Week, WorkWeek, Month, TimelineViews, TimelineMonth,
Agenda, MonthAgenda, Resize, DragAndDrop);
Calendar.Inject(Islamic);
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames, islamic);
let localeTexts: string;
let ajax: Ajax = new Ajax('./locale.json', 'GET', false);
ajax.onSuccess = (value: string) => {
    localeTexts = value;
};
ajax.send();
L10n.load(JSON.parse(<string>localeTexts));
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    enableRtl: true,
    calendarMode: 'Islamic',
    views: [
        { option: 'Day' },
        { option: 'TimelineDay' },
        { option: 'Week' },
        { option: 'TimelineWeek' },
        { option: 'Month' },
        { option: 'TimelineMonth' },
        { option: 'Agenda' },
        { option: 'MonthAgenda' }
    ],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
setCulture('ar');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Resources in EJ2 JavaScript Schedule control

Resources and grouping support allows the Scheduler to be shared by multiple resources. Also, the appointments of each resource are displayed under relevant resources. Each resource in the Scheduler is arranged in a column/row wise order, with individual spacing to display all its respective appointments on a single page. It also supports the multiple levels of grouping of resources, thus enabling the categorization of resources in a hierarchical structure and shows it either in expandable groups (Timeline views) or else vertical hierarchy one after the other (Calendar views).

It is also possible to assign one or more resources to the same appointment, by allowing multiple selection of resource options available in the event editor window.

The HTML5 JavaScript Scheduler groups the resources based on different criteria. It includes grouping appointments based on resources, grouping resources based on dates, and timeline scheduling. Also, the data for resources bind with Scheduler either as a local JSON collection or URL, retrieving data from remote data services.

Resource fields

The default options available within the `resources` collection are as follows,

Field name	Type	Description
------------	------	-------------

|-----|-----| ----- |

| **field** | String | A value that binds to the resource field of event object. |

| **title** | String | It holds the title of the resource field to be displayed on the event editor window. |

| **name** | String | A unique resource name used for differentiating various resource objects while grouping. |

| **allowMultiple** | Boolean | When set to **true**, allows multiple selection of resource names, thus creating multiple instances of same appointment for the selected resources. |

| **dataSource** | Object | Assigns the resource **dataSource**, where data can be passed either as an array of JavaScript objects, or else can create an instance of [DataManager](#) in case of processing remote data and can be assigned to the **dataSource** property. With the remote data assigned to **dataSource**, check the available [adaptors](#) to customize the data processing. |

| **query** | Query | Defines the external [query](#) that will be executed along with the data processing. |

| **idField** | String | Binds the resource ID field name from the resources **dataSource**. |

| **expandedField** | String | Binds the **expandedField** name from the resources **dataSource**. It usually holds boolean value which decide whether the resource of timeline views is in collapse or expand state on initial load. |

| **textField** | String | Binds the text field name from the resources **dataSource**. It usually holds the resource names. |

| **groupIDField** | String | Binds the group ID field name from the resource **dataSource**. It usually holds the value of resource IDs of parent level resources. |

| **colorField** | String | Binds the color field name from the resource **dataSource**. The color value mapped in this field will be applied to the events of resources. |

| **startHourField** | String | Binds the start hour field name from the resource **dataSource**. It allows to provide different work start hour for the resources. |

| **endHourField** | String | Binds the end hour field name from the resource **dataSource**. It allows to provide different work end hour for the resources. |

| **workDaysField** | String | Binds the work days field name from the resources **dataSource**. It allows to provide different working days collection for the resources. |

| **cssClassField** | String | Binds the custom CSS class field name from the resources **dataSource**. It maps the CSS class written for the specific resources and applies it to the events of those resources. |

Resource data binding

The data for resources can bind with Scheduler either as a local JSON collection or a service URL, retrieving resource data from remote data services.

Using local JSON data

The following code example depicts how to bind the local JSON data to the **dataSource** of **resources** collection.

```
`ts
```

```

import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject( Week, Month, TimelineViews, TimelineMonth, Agenda);
let scheduleObj: Schedule = new Schedule({
width: '100%',
height: '550px',
currentView: 'Week',
views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
selectedDate: new Date(2018, 3, 1),
resources: [{
field: 'OwnerId', title: 'Owner',
name: 'Owners', allowMultiple: true,
dataSource: [
{ OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
{ OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
{ OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
],
textField: 'OwnerText', idField: 'Id', colorField: 'OwnerColor'
}],
eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

Using remote service URL

The following code example depicts how to bind the remote data for resources `dataSource`.

```

`ts
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
let resource: DataManager = new DataManager({
url: 'Home/GetResourceData',
adaptor: new UrlAdaptor,

```

```

crossDomain: true
});
Schedule.Inject( Week, Month, TimelineViews, TimelineMonth, Agenda);
let scheduleObj: Schedule = new Schedule({
width: '100%',
height: '550px',
currentView: 'Week',
views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
selectedDate: new Date(2018, 3, 1),
resources: [{
field: 'OwnerId', title: 'Owner',
name: 'Owners', allowMultiple: true,
dataSource: resource,
textField: 'OwnerText', idField: 'Id', colorField: 'OwnerColor'
}],
eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

Scheduler with multiple resources

It is possible to display the Scheduler in default mode without visually showcasing all the resources in it, but allowing to assign the required resources to the appointments through the event editor resource options.

The appointments belonging to the different resources will be displayed altogether on the default Scheduler, which will be differentiated based on the resource color assigned in the **resources** (depicting to which resource that particular appointment belongs) collection.

Example: To display default Scheduler with multiple resource options in the event editor, ignore the group option and simply define the **resources** property with all its internal options.

INDEX.TS

```

import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject( Week, Month, TimelineViews, TimelineMonth, Agenda);
let scheduleObj: Schedule = new Schedule({
width: '100%',
height: '550px',
currentView: 'Week',
views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
selectedDate: new Date(2018, 3, 1),

```

```

resources: [{
    field: 'OwnerId', title: 'Owner',
    name: 'Owners', allowMultiple: true,
    dataSource: [
        { OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
        { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
        { OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
    ],
    textField: 'OwnerText', idField: 'Id', colorField: 'OwnerColor'
}],
eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```



```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting `allowMultiple` to `true` in the above code example allows you to select multiple resources from the event editor and also creates multiple copies of the same appointment in the Scheduler for each resources while rendering.

Resource grouping

Resource grouping support allows the Scheduler to group the resources in a hierarchical structure both as an expandable groups (Timeline views) and as vertical hierarchy displaying resources one after the other (Resources view).

Scheduler supports both single and multiple levels of resource grouping that can be customized both in timeline and vertical Scheduler views.

Vertical resource view

The following code example displays how the multiple resources are grouped and its events are portrayed in the default calendar views.

INDEX.TS

```

import { Schedule, Week, Month, Agenda } from '@syncfusion/ej2-schedule';
import { resourceConferenceData } from './datasource.ts';
Schedule.Inject( Week, Month, Agenda );
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    currentView: 'Week',
    views: ['Week', 'Month', 'Agenda'],
    selectedDate: new Date(2018, 3, 4),
    group: {
        byGroupID: false,
        resources: ['Projects', 'Categories']
    },
    resources: [
        {
            field: 'ProjectId', title: 'Choose Project', name: 'Projects',
            dataSource: [
                { text: 'PROJECT 1', id: 1, color: '#cb6bb2' },
                { text: 'PROJECT 2', id: 2, color: '#56ca85' },
                { text: 'PROJECT 3', id: 3, color: '#df5286' }
            ],
            textField: 'text', idField: 'id', colorField: 'color'
        }, {
            field: 'TaskId', title: 'Category',
            name: 'Categories', allowMultiple: true,
            dataSource: [
                { text: 'Development', id: 1, color: '#df5286' },
                { text: 'Testing', id: 2, color: '#7fa900' }
            ],
            textField: 'text', idField: 'id', colorField: 'color'
        }
    ],
},

```

```
eventSettings: { dataSource: resourceConferenceData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

[Timeline resource view](#)

The following code example depicts how to group the multiple resources on Timeline Scheduler views and its relevant events are displayed accordingly under those resources.

INDEX.TS

```
import { Schedule, TimelineViews, TimelineMonth, Agenda } from
 '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth, Agenda );
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    currentView: 'TimelineWeek',
    views: ['TimelineWeek', 'TimelineMonth', 'Agenda'],
    selectedDate: new Date(2018, 3, 4),
    group: {
        resources: ['Rooms', 'Owners']
    },
    resources: [{
        field: 'RoomId', title: 'Room',
        name: 'Rooms', allowMultiple: false,
        dataSource: [
            { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
            { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
        ],
        textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
    }, {
        field: 'OwnerId', title: 'Owner',
        name: 'Owners', allowMultiple: true,
        dataSource: [
            { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
 '#ffaa00' },
            { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
 '#f8a398' },
            { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor:
 '#7499e1' }
        ],
        textField: 'OwnerText', idField: 'Id', groupIDField: 'OwnerGroupId',
        colorField: 'OwnerColor'
    }],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping single-level resources

This kind of grouping allows the Scheduler to display all the resources at a single level simultaneously. The appointments mapped under resources will be displayed with the colors as per the `colorField` defined on the resources collection.

Example: To display the Scheduler with single level resource grouping,

INDEX.TS

```

import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Month, Week, Agenda, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
    selectedDate: new Date(2018, 3, 1),
    group: {
        resources: ['Owners']
    },
    resources: [{
        field: 'OwnerId', title: 'Owner',
        name: 'Owners', allowMultiple: true,

```

```

        dataSource: [
            { OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
            { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
            { OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
        ],
        textField: 'OwnerText', idField: 'Id', colorField: 'OwnerColor'
    }],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>

    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

The **name** field defined in the **resources** collection namely **Owners** will be mapped within the **group** property, in order to enable the grouping option with those resource levels on the Scheduler.

Grouping multi-level resources

It is possible to group the resources of Scheduler in multiple levels, by mapping the child resources to each parent resource. In the following example, there are 2 levels of resources, on which the second level resources are defined with **groupID** mapping to the first level resource's ID so as to establish the parent-child relationship between them.

Example: To display the Scheduler with multiple level resource grouping options,

INDEX.TS

```
import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from
 '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Month, Week, Agenda, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
  selectedDate: new Date(2018, 3, 1),
  group: {
    resources: ['Rooms', 'Owners']
  },
  resources: [{
    field: 'RoomId', title: 'Room',
    name: 'Rooms', allowMultiple: false,
    dataSource: [
      { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
      { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
    ],
    textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
  }, {
    field: 'OwnerId', title: 'Owner',
    name: 'Owners', allowMultiple: true,
    dataSource: [
      { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
        '#ffaa00' },
      { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
        '#f8a398' },
      { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor:
        '#7499e1' }
    ],
    textField: 'OwnerText', idField: 'Id', groupIDField: 'OwnerGroupId',
    colorField: 'OwnerColor'
  }],
  eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

One-to-One grouping

In multi-level grouping, Scheduler usually groups the resources on the child level based on the **GroupID** that maps with the **Id** field of parent level resources (as **byGroupID** set to true by default). There are also option which allows you to group all the child resource(s) against each of its parent resource(s). To enable this kind of grouping, set **false** to the **byGroupID** option within the **group** property. In the following code example, there are two levels of resources, on which all the 3 resources at the child level is mapped one to one with each resource on the first level.

INDEX.TS

```

import { Schedule, Week, Month, TimelineViews, TimelineMonth, Agenda } from
 '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Week, Month, TimelineViews, TimelineMonth, Agenda );
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    currentView: 'Week',
    views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
    selectedDate: new Date(2018, 3, 1),
    group: {
        byGroupID: false,
        resources: ['Rooms', 'Owners']
    },
    resources: [{
        field: 'RoomId', title: 'Room',
        name: 'Rooms', allowMultiple: false,
        dataSource: [
            { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
            { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
        ],
        textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
    }, {
        field: 'OwnerId', title: 'Owner',
        name: 'Owners', allowMultiple: true,
        dataSource: [
            { OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
            { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
            { OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
        ],
        textField: 'OwnerText', idField: 'Id', colorField: 'OwnerColor'
    }
    ],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grouping resources by date

It groups the number of resources under each date and is applicable only on the calendar views such as Day, Week, Work Week, Month, Agenda and Month-Agenda. To enable such grouping, set **byDate** option to **true** within the **group** property.

Example: To display the Scheduler with resources grouped by date,

INDEX.TS

```

import { Schedule, Week, Month, Agenda } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Month, Week, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Week', 'Month', 'Agenda'],
    selectedDate: new Date(2018, 3, 1),
    group: {
        byDate: true,
        resources: ['Owners']
    },
    resources: [{
        field: 'OwnerId', title: 'Owner',
        name: 'Owners', allowMultiple: true,
        dataSource: [
            { text: 'Alice', id: 1, color: '#1aaa55' },
            { text: 'Smith', id: 2, color: '#7fa900' }
        ]
    }],

```

```

        textField: 'text', idField: 'id', colorField: 'color'
    }],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

This kind of grouping by date is not applicable on any of the timeline views.

Customizing parent resource cells

In timeline view work cells of parent resource can be customized by checking the `elementType` as `resourceGroupCells` in the event `renderCell`. In the following code example, background color of the work hours has been changed.

INDEX.TS

```
import { Schedule, TimelineViews, TimelineMonth, RenderCellEventArgs } from
'@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  currentView: 'TimelineWeek',
  views: ['TimelineDay', 'TimelineWeek', 'TimelineMonth'],
  selectedDate: new Date(2018, 3, 4),
  group: {
    resources: ['Rooms', 'Owners']
  },
  resources: [{
    field: 'RoomId', title: 'Room',
    name: 'Rooms', allowMultiple: false,
    dataSource: [
      { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
      { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
    ],
    textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
  }, {
    field: 'OwnerId', title: 'Owner',
    name: 'Owners', allowMultiple: true,
    dataSource: [
      { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
'#ffaa00' },
      { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
'#f8a398' },
      { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor:
'#7499e1' }
    ],
    textField: 'OwnerText', idField: 'Id', groupIDField: 'OwnerGroupId',
colorField: 'OwnerColor'
  }],
  eventSettings: { dataSource: resourceData },
  renderCell: (args: RenderCellEventArgs) {
    if (args.elementType == 'resourceGroupCells' &&
args.element.className.indexOf('e-work-hours') > 0) {
      args.element.style.background = "#FAFAE3";
    }
  }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Working with shared events

Multiple resources can share the same events, thus allowing the CRUD action made on it to reflect on all other shared instances simultaneously. To enable such option, set `allowGroupEdit` option to `true` within the `group` property. With this property enabled, a single appointment object will be maintained within the appointment collection, even if it is shared by more than one resource – whereas the resource fields of such appointment object will be in array which hold the IDs of the multiple resources.

Any actions such as create, edit or delete held on any one of the shared event instances, will be reflected on all other related instances visible on the UI.

Example: To edit all the resource events simultaneously,

INDEX.TS

```

import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth, Resize } from '@syncfusion/ej2-schedule';
Schedule.Inject(Month, Week, Agenda, TimelineViews, TimelineMonth, Resize);
let resourceConferenceData: Object[] = [
    {
        Id: 1,
        Subject: 'Burning Man',
        StartTime: new Date(2018, 5, 1, 15, 0),
        EndTime: new Date(2018, 5, 1, 17, 0),
        ConferenceId: [1, 2, 3]
    }, {
        Id: 2,
        Subject: 'Data-Driven Economy',
        StartTime: new Date(2018, 5, 2, 12, 0),
        EndTime: new Date(2018, 5, 2, 14, 0),
        ConferenceId: [1, 2]
    }, {
        Id: 3,
        Subject: 'Techweek',
        StartTime: new Date(2018, 5, 2, 15, 0),
        EndTime: new Date(2018, 5, 2, 17, 0),
        ConferenceId: [2, 3]
    }, {
        Id: 4,
        Subject: 'Content Marketing World',
        StartTime: new Date(2018, 5, 2, 18, 0),
        EndTime: new Date(2018, 5, 2, 20, 0),
        ConferenceId: [1, 3]
    }, {
        Id: 5,
        Subject: 'B2B Marketing Forum',
        StartTime: new Date(2018, 5, 3, 10, 0),
        EndTime: new Date(2018, 5, 3, 12, 0),
        ConferenceId: [1, 2, 3]
    }
];
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 5, 5),
    views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
    currentView: 'Week',
    group: {
        allowGroupEdit: true,
        resources: ['Conferences']
    },
    resources: [{
        field: 'ConferenceId', title: 'Conference',
        name: 'Conferences', allowMultiple: true,
        dataSource: [
            { Text: 'Margaret', Id: 1, Color: '#1aaa55' },
            { Text: 'Robert', Id: 2, Color: '#357cd2' },
            { Text: 'Laura', Id: 3, Color: '#7fa900' }
        ],
        textField: 'Text', idField: 'Id', colorField: 'Color'
    }
],
});

```

```

    eventSettings: { dataSource: resourceConferenceData }
  });
  scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Simple resource header customization

It is possible to customize the resource header cells using built-in template option and change the look and appearance of it in both the vertical and timeline view modes. All the resource related fields and other information can be accessed within the resource header template option.

Example: To customize the resource header and display it along with designation field, refer the below code example.

INDEX.TS

```
import {
    Schedule, Week, Month, TimelineViews, TimelineMonth, Agenda
} from '@syncfusion/ej2-schedule';
import { doctorData } from './datasource.ts';
Schedule.Inject(Week, Month, TimelineViews, TimelineMonth, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 3, 1),
    resourceHeaderTemplate: '#resourceTemplate',
    views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
    group: {
        resources: ['Doctors']
    },
    resources: [{
        field: 'DoctorId', title: 'Doctor Name', name: 'Doctors',
        dataSource: [
            { text: 'Will Smith', id: 1, color: '#ea7a57', designation:
'Cardiologist' },
            { text: 'Alice', id: 2, color: '#7fa900', designation:
'Neurologist' },
            { text: 'Robson', id: 3, color: '#7fa900', designation:
'Orthopedic Surgeon' }
        ],
        textField: 'text', idField: 'id', colorField: 'color',
        designationField: 'designation'
    }],
    eventSettings: { dataSource: doctorData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
.e-schedule .e-vertical-view .e-resource-cells {
    height: 62px;
}
.e-schedule .template-wrap {
    display: flex;
    text-align: left;
}
.e-schedule .template-wrap .resource-details {
    padding-left: 10px;
}
.e-schedule .template-wrap .resource-details .resource-name {
    font-size: 16px;
    font-weight: 500;
    margin-top: 5px;
}
.e-schedule.e-device .template-wrap .resource-details .resource-name {
    font-size: inherit;
    font-weight: inherit;
}
.e-schedule.e-device .e-resource-tree-popup .e-fullrow {
    height: 50px;
}
.e-schedule.e-device .template-wrap .resource-details .resource-
designation {
    display: none;
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
    <script id="resourceTemplate" type="text/x-template">
        <div class='template-wrap'>
            <div class="resource-details">
                <div class="resource-name">${resourceData.text}</div>
                <div class="resource-
designation">${resourceData.designation}</div>
            </div>
        </div>
    </script>

    <div id="container">
        <div id="Schedule"></div>
    </div>

</script>

```

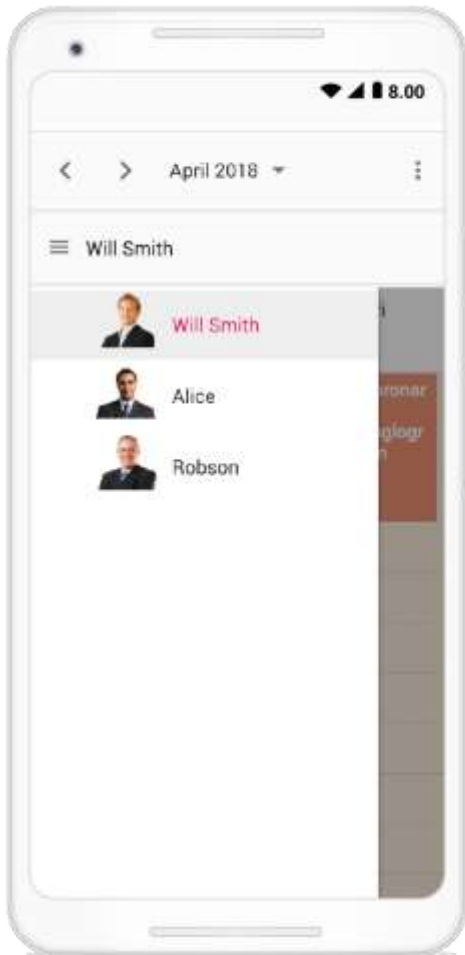


```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To customize the resource header in compact mode properly make use of the class `e-device` as in the code example.



Customizing resource header with multiple columns

It is possible to customize the resource headers to display with multiple columns such as Room, Type and Capacity. The following code example depicts the way to achieve it and is applicable only on timeline views.

INDEX.TS

```

import {
    Schedule, RenderCellEventArgs, TimelineViews, TimelineMonth
} from '@syncfusion/ej2-schedule';
import { roomData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth);

```

```

let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 7, 1),
    resourceHeaderTemplate: '#resourceTemplate',
    views: ['TimelineWeek', 'TimelineMonth'],
    group: {
        resources: ['MeetingRoom']
    },
    resources: [{
        field: 'RoomId', title: 'Room Type',
        name: 'MeetingRoom', allowMultiple: true,
        dataSource: [
            { text: 'Jammy', id: 1, color: '#ea7a57', capacity: 20, type:
'Conference' },
            { text: 'Tweety', id: 2, color: '#7fa900', capacity: 7, type:
'Cabin' },
            { text: 'Nestle', id: 3, color: '#5978ee', capacity: 5, type:
'Cabin' },
            { text: 'Phoenix', id: 4, color: '#fec200', capacity: 15, type:
'Conference' },
            { text: 'Mission', id: 5, color: '#df5286', capacity: 25, type:
'Conference' },
            { text: 'Hangout', id: 6, color: '#00bdae', capacity: 10, type:
'Cabin' },
            { text: 'Rick Roll', id: 7, color: '#865fcf', capacity: 20,
type: 'Conference' },
            { text: 'Rainbow', id: 8, color: '#1aaa55', capacity: 8, type:
'Cabin' },
            { text: 'Swarm', id: 9, color: '#df5286', capacity: 30, type:
'Conference' },
            { text: 'Photogenic', id: 10, color: '#710193', capacity: 25,
type: 'Conference' }
        ],
        textField: 'text', idField: 'id', colorField: 'color'
    }
    ],
    renderCell: (args: RenderCellEventArgs) => {
        if (args.elementType === 'emptyCells' &&
args.element.classList.contains('e-resource-left-td')) {
            let target: HTMLElement = args.element.querySelector('.e-
resource-text') as HTMLElement;
            target.innerHTML = '<div class="name">Rooms</div><div
class="type">Type</div><div class="capacity">Capacity</div>';
        }
    },
    eventSettings: { dataSource: roomData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
.e-schedule .e-timeline-view .e-resource-left-td,
.e-schedule .e-timeline-month-view .e-resource-left-td {
    vertical-align: bottom;
}
.e-schedule.e-device .e-timeline-view .e-resource-left-td,
.e-schedule.e-device .e-timeline-month-view .e-resource-left-td {
    width: 75px;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text,
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text
{
    display: flex;
    font-weight: 500;
    padding: 0;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-text>div,
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div {
    border-right: 1px solid rgba(0, 0, 0, 0.12);
    border-top: 1px solid rgba(0, 0, 0, 0.12);
    flex: 0 0 33.3%;
    font-weight: 500;
    height: 36px;
    line-height: 34px;
    padding-left: 5px;
}
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div {
    border-top: 0;
}
.e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div:last-child,
.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div:last-child {
    border-right: 0;
}

```

```

.e-schedule .template-wrap {
  display: flex;
  height: 100%;
  text-align: left;
}
.e-schedule .template-wrap>div {
  border-right: 1px solid rgba(0, 0, 0, 0.12);
  flex: 0 0 33.3%;
  font-weight: 500;
  line-height: 58px;
  overflow: hidden;
  padding-left: 5px;
  text-overflow: ellipsis;
}
.e-schedule .template-wrap>div:last-child {
  border-right: 0;
}
.e-schedule .e-timeline-view .e-resource-cells,
.e-schedule .e-timeline-month-view .e-resource-cells {
  padding-left: 0;
}
.e-schedule .e-timeline-view .e-date-header-wrap table col,
.e-schedule .e-timeline-view .e-content-wrap table col {
  width: 100px;
}
@media (max-width: 550px) {
  .e-schedule .e-timeline-view .e-resource-left-td,
  .e-schedule .e-timeline-month-view .e-resource-left-td {
    width: 100px;
  }
  .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div,
  .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div,
  .e-schedule .template-wrap>div {
    flex: 0 0 100%;
  }
  .e-schedule .template-wrap>div:first-child {
    border-right: 0;
  }
  .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div:first-child,
  .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div:first-child {
    border-right: 0;
  }
  .e-schedule .room-type,
  .e-schedule .room-capacity {
    display: none;
  }
}
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head><body>
  <script id="resourceTemplate" type="text/x-template">
    <div class='template-wrap'>
      <div class="room-name">${resourceData.text}</div>
      <div class="room-type">${resourceData.type}</div>
      <div class="room-capacity">${resourceData.capacity}</div>
    </div>
  </script>

  <div id="container">
    <div id="Schedule"></div>
  </div>

  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Collapse/Expand child resources in timeline views

It is possible to expand and collapse the resources which have child resource in timeline views dynamically. By default, resources are in expanded state with their child resource. We can collapse and expand the child resources in UI by setting `expandedField` option as `false` whereas its default value is `true`.

INDEX.TS

```

import { Schedule, TimelineViews, TimelineMonth, Agenda } from
'@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth, Agenda);
let scheduleObj: Schedule = new Schedule({
  width: "100%",
  height: "550px",
  currentView: "TimelineWeek",
  views: ["TimelineWeek", "TimelineMonth", "Agenda"],
  selectedDate: new Date(2018, 3, 4),
  group: {
    resources: ["Rooms", "Owners"]
  },
  resources: [
    {
      field: "RoomId", title: "Room",
      name: "Rooms", allowMultiple: false,
      dataSource: [
        { RoomText: "ROOM 1", Id: 1, RoomColor: "#cb6bb2", IsExpand: false
      },
        { RoomText: "ROOM 2", Id: 2, RoomColor: "#56ca85" }
      ],
      textField: "RoomText", idField: "Id", colorField: "RoomColor",
      expandedField: "IsExpand"
    },
  ],

```

```

{
    field: "OwnerId", title: "Owner",
    name: "Owners", allowMultiple: true,
    dataSource: [
        { OwnerText: "Nancy", Id: 1, OwnerGroupId: 1, OwnerColor: "#ffaa00"
    },
        { OwnerText: "Steven", Id: 2, OwnerGroupId: 2, OwnerColor: "#f8a398"
    },
        { OwnerText: "Michael", Id: 3, OwnerGroupId: 1, OwnerColor:
"#7499e1" },
        { OwnerText: "Alice", Id: 4, OwnerGroupId: 2, OwnerColor: "#7fa900"
    }
    ],
    textField: "OwnerText", idField: "Id", groupIDField: "OwnerGroupId",
    colorField: "OwnerColor"
}
],
eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo("#Schedule");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

        <div id="container">
            <div id="Schedule"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Displaying tooltip for resource headers

It is possible to display tooltips over the resource headers showing the resource information. By default, there won't be any tooltips displayed on the resource headers, and to enable it, you need to assign the customized template design to the `headerTooltipTemplate` option within the `group` property.

INDEX.TS

```

import {
    Schedule, Week, Month, TimelineViews, TimelineMonth
} from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Week, Month, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 3, 1),
    views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth'],
    group: {
        resources: ['Rooms', 'Owners'],
        headerTooltipTemplate: '#tooltipTemplate'
    },
    resources: [
        {
            field: 'OwnerId', title: 'Owner',
            name: 'Owners', allowMultiple: true,
            dataSource: [
                { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
                    '#ffaa00' },
                { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
                    '#f8a398' },
                { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor:
                    '#7499e1' }
            ],
            textField: 'OwnerText', idField: 'Id', groupIDField:
                'OwnerGroupId', colorField: 'OwnerColor'
        }
    ],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-schedule .e-vertical-view .e-resource-cells {
      height: 45px;
    }

    .e-schedule .e-agenda-view .template-wrap .resource-text {
      text-align: center;
    }

    .e-schedule .template-wrap .resource-text {
      font-size: 15px;
      padding: 4px 4px 4px;
      height: 25px;
      text-overflow: ellipsis;
      white-space: nowrap;
      overflow: hidden;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head><body>
  <script id="tooltipTemplate" type="text/x-template">
    <div class='template-wrap'>
      <div class="res-text">Name:  ${resourceData.OwnerText} </div>
    </div>
  </script>

  <div id="container">
    <div id="Schedule"></div>
  </div>

```



```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Choosing among resource colors for appointments

By default, the colors defined on the top level resources collection will be applied for the events. In case, if you want to apply specific resource color to events irrespective of its top-level parent resource color, it can be achieved by defining **resourceColorField** option within the **eventSettings** property.

In the following example, the colors mentioned in the second level will get applied over the events.

INDEX.TS

```
import { RadioButton, ChangeArgs } from '@syncfusion/ej2-buttons';
import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Month, Week, Agenda, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
    selectedDate: new Date(2018, 3, 1),
    group: {
        resources: ['Rooms', 'Owners']
    },
    resources: [{
        field: 'RoomId', title: 'Room',
        name: 'Rooms', allowMultiple: false,
        dataSource: [
            { RoomText: 'ROOM 1', Id: 1, RoomGroupId: 1, RoomColor: '#cb6bb2' },
            { RoomText: 'ROOM 2', Id: 2, RoomGroupId: 2, RoomColor: '#56ca85' }
        ],
        textField: 'RoomText', idField: 'Id', groupIDField: 'RoomGroupId',
        colorField: 'RoomColor'
    }, {
        field: 'OwnerId', title: 'Owner',
        name: 'Owners', allowMultiple: true,
        dataSource: [
            { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor: '#ffaa00' },
            { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor: '#f8a398' },
            { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor: '#7499e1' }
        ],
        textField: 'OwnerText', idField: 'Id', groupIDField: 'OwnerGroupId',
        colorField: 'OwnerColor'
    }
]},
```

```

    eventSettings: { dataSource: resourceData, resourceColorField: 'Rooms' }
  });
  scheduleObj.appendTo('#Schedule');
  new RadioButton({ value: 'Rooms', name: 'default', label: 'Rooms', checked:
  true, change: onChange }, '#room');
  new RadioButton({ value: 'Owners', name: 'default', label: 'Owners',
  checked: false, change: onChange }, '#owner');
  function onChange(args: ChangeArgs): void {
    scheduleObj.eventSettings.resourceColorField = args.value;
  }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div class="col-lg-12 property-section">
    <table id="property" title="Use Color of Room / Owner">
      <tbody>
        <tr>
          <td>
            <input id="room" class="room" type="radio">
          </td>
          <td>
            <input id="owner" class="owner" type="radio">
          </td>
        </tr>
      </tbody>
    </table>
  </div>

```

```

        </tr>
      </tbody>
    </table>
  </div>
  <div id="container">
    <div id="Schedule"></div>
  </div>
  <style>
    #property td {
      padding: 0 5px;
    }
  </style>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The value of the `resourceColorField` field should be mapped with the `name` value given within the `resources` property.

Setting different style to each resource appointments

By default, the appearance of events is the same for all resource events. In case, if you want to apply the different styles to each resource event, you can do this by defining the `cssClassField` option within the `resource` property that maps the different `cssClass` fields from the resource `dataSource` as depicted in the following example.

INDEX.TS

```

import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(Month, Week, Agenda, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
  selectedDate: new Date(2018, 3, 1),
  group: {
    resources: ['Owners']
  },
  resources: [
    {
      field: 'OwnerId', title: 'Owner',
      name: 'Owners', allowMultiple: true,
      dataSource: [
        { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
'#ffaa00', ResourceCssClass: "NancyResource" },
        { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
'#f8a398', ResourceCssClass: "StevenResource" },
        { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor:
'#7499e1', ResourceCssClass: "MichaelResource" }
      ]
    }
  ]
});

```

```

    ],
    textField: 'OwnerText', idField: 'Id', groupIDField: 'OwnerGroupId',
    colorField: 'OwnerColor', cssClassField: "ResourceCssClass"
  }],
  eventSettings: { dataSource: resourceData, resourceColorField: 'Rooms' }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <style>
    .e-appointment.NancyResource {
      border-bottom: 3px solid red !important;
    }
    .e-appointment.StevenResource {
      border-radius: 5px !important;
      border-bottom: 3px solid green !important;
    }
    .e-appointment.MichaelResource {
      border: 1px solid black !important;
    }
  </style>

```

```

    }
    </style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Dynamically add and remove resources

It is possible to add or remove the resources dynamically to and from the Scheduler respectively. In the following example, when the checkboxes are checked and unchecked, the respective resources gets added up or removed from the Scheduler layout. To add new resource dynamically, **addResource** method is used which accepts the arguments such as resource object, resource name (within which level, the resource object to be added) and index (position where the resource needs to be added).

To remove the resources dynamically, **removeResource** method is used which accepts the index (position from where the resource to be removed) and resource name (within which level, the resource object presents) as parameters.

INDEX.TS

```

import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
import { Schedule, Month, TimelineMonth } from '@syncfusion/ej2-schedule';
import { holidayData, birthdayData, companyData, personalData } from
'./datasource.ts';
Schedule.Inject(Month, TimelineMonth);
let calendarCollections: Object[] = [
    { CalendarText: 'My Calendar', CalendarId: 1, CalendarColor: '#c43081'
},
    { CalendarText: 'Company', CalendarId: 2, CalendarColor: '#ff7f50' },
    { CalendarText: 'Birthday', CalendarId: 3, CalendarColor: '#AF27CD' },
    { CalendarText: 'Holiday', CalendarId: 4, CalendarColor: '#808000' }
];
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 3, 1),
    group: {
        resources: ['Calendars']
    },
    resources: [{
        field: 'CalendarId', title: 'Calendar',
        name: 'Calendars', allowMultiple: true,
        dataSource: [calendarCollections[0]],
        textField: 'CalendarText', idField: 'CalendarId', colorField:
'CalendarColor'
    }],
    views: ['Month', 'TimelineMonth'],
    eventSettings: { dataSource: generateCalendarData() }
});
scheduleObj.appendTo('#Schedule');
function onChange(args: ChangeEventArgs): void {

```

```

    let value: number =
    parseInt((<Element>args.event.target).getAttribute('value'), 10);
    let resourceData: Object[] = calendarCollections.filter((calendar: {
    [key: string]: Object }) => calendar.CalendarId === value);
    if (args.checked) {
        scheduleObj.addResource(resourceData[0], 'Calendars', value - 1);
    } else {
        scheduleObj.removeResource(value, 'Calendars');
    }
}
new CheckBox({ value: '1', label: 'My Calendar', checked: true, disabled:
true, change: onChange }, '#personal');
new CheckBox({ value: '2', label: 'Company', checked: false, change:
onChange }, '#company');
new CheckBox({ value: '3', label: 'Birthday', checked: false, change:
onChange }, '#birthdays');
new CheckBox({ value: '4', label: 'Holiday', checked: false, change:
onChange }, '#holidays');
function generateCalendarData(): Object[] {
    let collections: Object[] = [];
    let dataCollections: Object[][] = [personalData, companyData,
birthdayData, holidayData];
    for (let data of dataCollections) {
        collections = collections.concat(data);
    }
    return collections;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-lg-12 property-section">
        <table id="property" title="Show / Hide Resource">
            <tbody>
                <tr>
                    <td>
                        <input id="personal" class="e-resource-calendar e-
personal" type="checkbox">
                    </td>
                    <td>
                        <input id="company" class="e-resource-calendar e-
company" type="checkbox">
                    </td>
                    <td>
                        <input id="birthdays" class="e-resource-calendar e-
birthday" type="checkbox">
                    </td>
                    <td>
                        <input id="holidays" class="e-resource-calendar e-
holiday" type="checkbox">
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
    <div id="container">
        <div id="Schedule"></div>
    </div>
    <style>
        #property td {
            padding: 0 5px;
        }
        .e-schedule .e-timeline-view .e-resource-left-td,
        .e-schedule .e-timeline-month-view .e-resource-left-td {
            width: 110px;
        }
    </style>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting different working days and hours for resources

Each resource in the Scheduler can have different working hours as well as different working days set to it. There are default options available within the `resources` collection, to customize the default working hours and days of the Scheduler.

Set different work days

Different working days can be set for the resources of Scheduler using the `workDaysField` property which maps the working days field from the resource dataSource. This field accepts the collection of day indexes (from 0 to 6) of a week. By default, it is set to [1, 2, 3, 4, 5] and in the following example, each resource has been set with different values and therefore each of them will render only those working days. This option is applicable only on the calendar views and is not applicable on timeline views.

INDEX.TS

```
import { Schedule, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { doctorData } from './datasource.ts';
Schedule.Inject(Month, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 3, 1),
    views: ['Week', 'WorkWeek', 'Month'],
    currentView: 'WorkWeek',
    group: {
        resources: ['Doctors']
    },
    resources: [{
        field: 'DoctorId', title: 'Choose Doctor',
        name: 'Doctors', allowMultiple: true,
        dataSource: [
            { text: 'Will Smith', id: 1, color: '#ea7a57', workDays: [1, 2, 4, 5] },
            { text: 'Alice', id: 2, color: 'rgb(53, 124, 210)', workDays: [1, 3, 5] },
            { text: 'Robson', id: 3, color: '#7fa900', workDays: [2, 6] }
        ],
        textField: 'text', idField: 'id', colorField: 'color',
        workDaysField: 'workDays'
    }],
    eventSettings: { dataSource: doctorData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set different work hours

Working hours indicates the work hour duration of a day, which is highlighted visually with active color over the work cells. Each resource on the Scheduler can be defined with its own set of working hours as depicted in the following example.

- **startHourField** - Denotes the start time of the working/business hour in a day.
- **endHourField** - Denotes the end time limit of the working/business hour in a day.

INDEX.TS

```

import { Schedule, Week, Month, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-schedule';
import { doctorData } from './datasource.ts';
Schedule.Inject(Week, Month, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 3, 1),

```

```

views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth'],
group: {
  resources: ['Doctors']
},
resources: [{
  field: 'DoctorId', title: 'Doctor Name',
  name: 'Doctors', allowMultiple: true,
  dataSource: [
    { text: 'Will Smith', id: 1, color: '#ea7a57', startHour:
'08:00', endHour: '15:00' },
    { text: 'Alice', id: 2, color: '#357cd2', startHour: '10:00',
endHour: '18:00' },
    { text: 'Robson', id: 3, color: '#7fa900', startHour: '06:00',
endHour: '13:00' }
  ],
  textField: 'text', idField: 'id', colorField: 'color',
  startHourField: 'startHour', endHourField: 'endHour'
}],
eventSettings: { dataSource: doctorData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In this example, a resource named **Will Smith** is depicted with working hours ranging from 8.00 AM to 3.00 PM and is visually illustrated with active colors, whereas the other two resources have different working hours set.

Hide non-working days when grouped by date

In Scheduler, you can set custom work days for each resource and group the Scheduler by date to display these work days. By default, the Scheduler will show all days when it is grouped by date, even if they are not included in the custom work days for the resources. However, you can use the [hideNonWorkingDays](#) property to only display the custom work days in the Scheduler.

To use the [hideNonWorkingDays](#) property, you need to include it in the configuration options for your Scheduler component. Set the value of [hideNonWorkingDays](#) to **true** to enable this feature.

Example: To display the Scheduler with resources grouped by date for custom working days,

INDEX.TS

```

import { Schedule, ScheduleModel, Day, Week, Month, Agenda } from
'@syncfusion/ej2-schedule';
import * as dataSource from './datasource.json';
/**
 * schedule resources group sample
 */
Schedule.Inject(Day, Week, Month, Agenda);
let scheduleOptions: ScheduleModel = {
  width: '100%',
  height: '650px',
  group: {
    byDate: true,
    hideNonWorkingDays: true,
    resources: ['Owners']
  },
  resources: [{
    field: 'TaskId', title: 'Assignee',
    name: 'Owners', allowMultiple: true,
    dataSource: [
      { text: 'Alice', id: 1, color: '#df5286', workDays: [1, 2,
3, 4] },
      { text: 'Smith', id: 2, color: '#7fa900', workDays: [2, 3,
5] }
    ],
    textField: 'text', idField: 'id', colorField: 'color',
    workDaysField: 'workDays'
  }
]
}

```

```

    }],
  };
  let scheduleObj: Schedule = new Schedule(scheduleOptions);
  scheduleObj.appendTo(document.getElementById('Schedule'));

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The [hideNonWorkingDays](#) property only applies when the Scheduler is grouped [byDate](#).

Scroll to specific resource

You can manually scroll to a specific resource on Scheduler by making use of the `scrollToResource` method as depicted in the following code example.

INDEX.TS

```
import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth } from
 '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
import { resourceData } from './datasource.ts';
Schedule.Inject(Month, Week, Agenda, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  views: ['Week', 'Month', 'TimelineWeek', 'TimelineMonth', 'Agenda'],
  selectedDate: new Date(2018, 3, 1),
  group: {
    resources: ['Owners']
  },
  resources: [
    {
      field: 'OwnerId', title: 'Owner',
      name: 'Owners', allowMultiple: true,
      dataSource: [
        { OwnerText: 'Jammy', Id: 1, OwnerColor: '#ffaa00' },
        { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
        { OwnerText: 'Tweety', Id: 3, OwnerColor: '#7499e1' },
        { OwnerText: 'Jammy', Id: 4, OwnerColor: '#ffaa00' },
        { OwnerText: 'Nestle', Id: 5, OwnerGroupId: 2, OwnerColor:
 '#f8a398' },
        { OwnerText: 'Phoenix', Id: 6, OwnerColor: '#7499e1' },
        { OwnerText: 'Hangout', Id: 7, OwnerColor: '#ffaa00' },
        { OwnerText: 'Rainbow', Id: 8, OwnerGroupId: 2, OwnerColor:
 '#f8a398' },
        { OwnerText: 'Photogenic', Id: 9, OwnerColor: '#7499e1' }
      ],
      textField: 'OwnerText', idField: 'Id', colorField: 'OwnerColor'
    }
  ],
  eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');
let button1: Button = new Button();
button1.appendTo('#btn1');
button1.element.onclick = (): void => {
  scheduleObj.scrollToResource(6, 'Owners');
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-lg-12 property-section">
        <button id="btn1"> Click Me to Navigate Phoenix </button>
    </div>
    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Compact view in mobile

Although the Scheduler views are designed keeping in mind the responsiveness of the control in mobile devices, however when using Scheduler with multiple resources - it is difficult to view all the resources and its relevant events at once on the mobile. Therefore, we have introduced a new compact mode specially for displaying multiple resources of Scheduler on mobile devices. By default, this mode is enabled while using Scheduler with multiple resources on mobile devices. If in case, you need to disable this compact mode, set `false` to the `enableCompactView` option within the `group` property. Disabling this option will display the exact desktop mode of Scheduler view on mobile devices.

With this compact view enabled on mobile, you can view only single resource at a time and to switch to other resources, there is a treeview at the left listing out all other available resources - clicking on which will display that particular resource and its related appointments.



Clicking on the menu icon before the resource text will show the resources available in the Scheduler as following.



Adaptive UI in desktop

By default, the Scheduler layout adapts automatically in the desktop and mobile devices with appropriate UI changes. In case, if the user wants to display the Adaptive scheduler in desktop mode with adaptive enhancements, then the property `enableAdaptiveUI` can be set to true. Enabling this option will display the exact mobile mode of Scheduler view on desktop devices.

Some of the default changes made for compact Scheduler to render in desktop devices are as follows,

- View options displayed in the Navigation drawer.
- Plus icon is added to the header for new event creation.
- Today icon is added to the header instead of the Today button.
- With Multiple resources – only one resource has been shown to enhance the view experience of resource events details clearly. To switch to other resources, there is a TreeView on the left that lists all other available resources, clicking on which will display that particular resource and its related events.

INDEX.TS

```
import { Schedule, Day, Week, Month, Resize, DragAndDrop } from
'@syncfusion/ej2-schedule';
import { extend } from '@syncfusion/ej2-base';
import { timelineResourceData, resourceData } from './datasource.ts';
```



```

Schedule.Inject(Day, Week, Month, Resize, DragAndDrop);
let scheduleObj: Schedule = new Schedule({
  width: '100%', height: '555px',
  views: ['Day', 'Week', 'Month'],
  currentView: 'Month',
  selectedDate: new Date(2018, 3, 4),
  enableAdaptiveUI: true,
  group: {
    resources: ['Projects', 'Categories']
  },
  resources: [
    {
      field: 'ProjectId', title: 'Choose Project', name: 'Projects',
      dataSource: [
        { text: 'PROJECT 1', id: 1, color: '#cb6bb2' },
        { text: 'PROJECT 2', id: 2, color: '#56ca85' },
        { text: 'PROJECT 3', id: 3, color: '#df5286' }
      ],
      textField: 'text', idField: 'id', colorField: 'color'
    }, {
      field: 'TaskId', title: 'Category',
      name: 'Categories', allowMultiple: true,
      dataSource: [
        { text: 'Nancy', id: 1, groupId: 1, color: '#df5286' },
        { text: 'Steven', id: 2, groupId: 1, color: '#7fa900' },
        { text: 'Robert', id: 3, groupId: 2, color: '#ea7a57' },
        { text: 'Smith', id: 4, groupId: 2, color: '#5978ee' },
        { text: 'Michael', id: 5, groupId: 3, color: '#df5286' },
        { text: 'Root', id: 6, groupId: 3, color: '#00bdae' }
      ],
      textField: 'text', idField: 'id', groupIDField: 'groupId',
      colorField: 'color'
    }
  ],
  eventSettings: {
    dataSource: <Object[]> extend([],
      resourceData.concat(timelineResourceData), null, true)
  }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Header rows in EJ2 JavaScript Schedule control

The Timeline views can have additional header rows other than its default date and time header rows. It is possible to show individual header rows for displaying year, month and week separately using the `headerRows` property. This property is applicable only on the timeline views. The possible rows which can be added using `headerRows` property are as follows.

- Year
- Month
- Week
- Date
- Hour

The `Hour` row is not applicable for Timeline month view.

The following example shows the Scheduler displaying all the available header rows on timeline views.

INDEX.TS

```
import { Schedule, TimelineViews } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews);
let scheduleObj: Schedule = new Schedule({
    width: '100%', height: '550px',
    selectedDate: new Date(2018, 11, 31),
    startHour: '09:00',
    endHour: '13:00',
    headerRows: [
        { option: 'Year' },
        { option: 'Month' },
        { option: 'Week' },
        { option: 'Date' },
        { option: 'Hour' }
    ],
    views: ['TimelineWeek'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
```

```

        <div id="Schedule"></div>
    </div>
    <script id="year-template" type="text/x-template">
        <span class="year">${getYearDetails(data)}</span>
    </script>
    <script id="month-template" type="text/x-template">
        <span class="month">${getMonthDetails(data)}</span>
    </script>
    <script id="week-template" type="text/x-template">
        <span class="week">${getWeekDetails(data)}</span>
    </script>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Display year and month rows in timeline views

To display the timeline Scheduler simply with year and month names alone, define the option **Year** and **Month** within the **headerRows** property.

INDEX.TS

```

import { Schedule, TimelineMonth } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%', height: '550px',
    selectedDate: new Date(2018, 11, 31),
    headerRows: [{ option: 'Year' }, { option: 'Month' }],
    currentView: 'TimelineMonth',
    views: [{ option: 'TimelineMonth', interval: 24 }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script id="year-template" type="text/x-template">
        <span class="year">${getYearDetails(data)}</span>
    </script>
    <script id="month-template" type="text/x-template">
        <span class="month">${getMonthDetails(data)}</span>
    </script>
    <script id="week-template" type="text/x-template">
        <span class="week">${getWeekDetails(data)}</span>
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Display week numbers in timeline views

The week number can be displayed in a separate header row of the timeline Scheduler by setting **Week** option within **headerRows** property.

INDEX.TS

```

import { Schedule, TimelineViews, TimelineMonth } from '@syncfusion/ej2-
schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%', height: '550px',
    selectedDate: new Date(2018, 1, 15),
    headerRows: [{ option: 'Week' }, { option: 'Date' }, { option: 'Hour'
}],

```

```

    currentView: 'TimelineMonth',
    views: [{ option: 'TimelineMonth', interval: 24 }, { option:
'TimelineWeek', interval: 3 }, { option: 'TimelineDay', interval: 4 }],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script id="year-template" type="text/x-template">
    <span class="year">${getYearDetails(data)}</span>
  </script>
  <script id="month-template" type="text/x-template">
    <span class="month">${getMonthDetails(data)}</span>
  </script>
  <script id="week-template" type="text/x-template">
    <span class="week">${getWeekDetails(data)}</span>
  </script>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline view displaying dates of a complete year

It is possible to display a complete year in a timeline view by setting **interval** value as 12 and defining **TimelineMonth** view option within the **views** property of Scheduler.

INDEX.TS

```

import { Schedule, TimelineMonth } from '@syncfusion/ej2-schedule';
import { eventData } from './datasource.ts';
Schedule.Inject(TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%', height: '550px',
    selectedDate: new Date(2018, 0, 1),
    headerRows: [{ option: 'Month' }, { option: 'Date' }],
    views: [{ option: 'TimelineMonth', interval: 12 }],
    eventSettings: { dataSource: eventData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script id="year-template" type="text/x-template">
        <span class="year">${getYearDetails(data)}</span>
    </script>
    <script id="month-template" type="text/x-template">
        <span class="month">${getMonthDetails(data)}</span>
    </script>
    <script id="week-template" type="text/x-template">
        <span class="week">${getWeekDetails(data)}</span>
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the header rows using template

You can customize the text of the header rows and display any images or formatted text on each individual header rows using the built-in **template** option available within the **headerRows** property.

INDEX.TS

```

import { Schedule, TimelineMonth, CellTemplateArgs, getWeekNumber } from
'@syncfusion/ej2-schedule';
import { Internationalization } from '@syncfusion/ej2-base';
import { eventData } from './datasource.ts';
Schedule.Inject(TimelineMonth);
interface TemplateFunction extends Window {
    getYearDetails?: Function;
    getMonthDetails?: Function;
    getWeekDetails?: Function;
}
let instance: Internationalization = new Internationalization();
(window as TemplateFunction).getYearDetails = (value: CellTemplateArgs) => {
    return 'Year: ' + instance.formatDate((value as CellTemplateArgs).date,
    { skeleton: 'Y' });
};
(window as TemplateFunction).getMonthDetails = (value: CellTemplateArgs) => {
    return 'Month: ' + instance.formatDate((value as CellTemplateArgs).date,
    { skeleton: 'M' });
};
(window as TemplateFunction).getWeekDetails = (value: CellTemplateArgs) => {
    return 'Week ' + getWeekNumber((value as CellTemplateArgs).date);
};
let scheduleObj: Schedule = new Schedule({

```



```

width: '100%',
height: '550px',
selectedDate: new Date(2018, 0, 1),
headerRows: [
    { option: 'Year', template: '#year-template' },
    { option: 'Month', template: '#month-template' },
    { option: 'Week', template: '#week-template' },
    { option: 'Date' }
],
views: [{ option: 'TimelineMonth' }],
eventSettings: { dataSource: eventData }
};
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script id="year-template" type="text/x-template">
        <span class="year">${getYearDetails(data)}</span>
    </script>
    <script id="month-template" type="text/x-template">

```

```

        <span class="month">${getMonthDetails(data)}</span>
    </script>
    <script id="week-template" type="text/x-template">
        <span class="week">${getWeekDetails(data)}</span>
    </script>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Row auto height in EJ2 JavaScript Schedule control

By default, the height of the Scheduler rows in Timeline views are static and therefore, when the same time range holds multiple overlapping appointments, a **+n more** text indicator will be displayed. With this feature enabled, you can now view all the overlapping appointments present in those specific time range by auto-adjusting the row height based on the presence of the appointments count, instead of displaying the **+n more** text indicators.

To enable auto row height adjustments on Scheduler Timeline views and Month view, set **true** to the **rowAutoHeight** property whose default value is **false**.

This auto row height adjustment is applicable only on all the Timeline views as well as on the calendar Month view.

Now, let's see how it works on those applicable views with examples.

Calendar month view

By default, the rows of the calendar Month view can hold only the limited appointments count based on its row height, and the rest of the overlapping appointments are indicated with a **+n more** text indicator. The following example shows how the month view row auto-adjusts based on the number of appointments count, when this **rowAutoHeight** feature is enabled.

INDEX.TS

```

import { Schedule, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '450px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'Month',
    rowAutoHeight: true,
    views: ['Month'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline views

When the feature **rowAutoHeight** is enabled in Timeline views, the row height gets auto-adjusted based on the number of overlapping events occupied on the same time range, which is demonstrated in the following example.

INDEX.TS

```
import { Schedule, TimelineViews, TimelineMonth, Agenda, Resize, DragAndDrop
} from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth, Agenda, Resize, DragAndDrop);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '450px',
    rowAutoHeight: true,
    selectedDate: new Date(2018, 1, 15),
    currentView: 'TimelineWeek',
    views: [
        { option: 'TimelineDay' },
        { option: 'TimelineWeek' },
        { option: 'TimelineWorkWeek' },
        { option: 'TimelineMonth' },
        { option: 'Agenda' }
    ],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <div id="Schedule"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Timeline views with multiple resources

The following example shows how the auto row adjustment feature works on timeline views with multiple resources.

INDEX.TS

```

import { Schedule, TimelineViews, Resize, DragAndDrop } from
 '@syncfusion/ej2-schedule';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
import { roomData } from './datasource.ts';
Schedule.Inject(TimelineViews, Resize, DragAndDrop);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 7, 1),
  currentView: 'TimelineWeek',
  rowAutoHeight: true,
  views: ['TimelineDay', 'TimelineWeek'],
  group: {
    enableCompactView: false,
    resources: ['MeetingRoom']
  },
  resources: [{
    field: 'RoomId', title: 'Room Type',
    name: 'MeetingRoom', allowMultiple: true,
    dataSource: [
      { text: 'Room A', id: 1, color: '#98AFC7' },
      { text: 'Room B', id: 2, color: '#99c68e' },
      { text: 'Room C', id: 3, color: '#C2B280' },
      { text: 'Room D', id: 4, color: '#3090C7' },
      { text: 'Room E', id: 5, color: '#95b9' },
      { text: 'Room F', id: 6, color: '#95b9c7' },
      { text: 'Room G', id: 7, color: '#deb887' },
      { text: 'Room H', id: 8, color: '#3090C7' },
      { text: 'Room I', id: 9, color: '#98AFC7' },
      { text: 'Room J', id: 10, color: '#778899' }
    ],
    textField: 'text', idField: 'id', colorField: 'color'
  }],
  eventSettings: {
    dataSource: roomData,
    fields: {
      id: 'Id',
      subject: { title: 'Summary', name: 'Subject' },

```

```

        location: { title: 'Location', name: 'Location' },
        description: { title: 'Comments', name: 'Description' },
        startTime: { title: 'From', name: 'StartTime' },
        endTime: { title: 'To', name: 'EndTime' }
    }
}
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Appointments occupying entire cell

By default, with the feature `rowAutoHeight`, there will be a space in the bottom of the cell when appointment is rendered. To avoid this space, we can set true to the property `ignoreWhitespace` with in `eventSettings` whereas its default property value is false. In the following code example, the whitespace below the appointments has been ignored.

INDEX.TS

```
import { Schedule, TimelineViews, TimelineMonth } from '@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject(TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  currentView: 'TimelineWeek',
  rowAutoHeight: true,
  views: ['TimelineWeek', 'TimelineMonth'],
  selectedDate: new Date(2021, 7, 4),
  group: {
    resources: ['Rooms', 'Owners']
  },
  resources: [{
    field: 'RoomId', title: 'Room',
    name: 'Rooms', allowMultiple: false,
    dataSource: [
      { RoomText: 'ROOM 1', Id: 1, RoomColor: '#cb6bb2' },
      { RoomText: 'ROOM 2', Id: 2, RoomColor: '#56ca85' }
    ],
    textField: 'RoomText', idField: 'Id', colorField: 'RoomColor'
  }, {
    field: 'OwnerId', title: 'Owner',
    name: 'Owners', allowMultiple: true,
    dataSource: [
      { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor: '#ffaa00' },
      { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor: '#f8a398' },
      { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor: '#7499e1' }
    ],
    textField: 'OwnerText', idField: 'Id', groupIDField: 'OwnerGroupId',
    colorField: 'OwnerColor'
  }],
  eventSettings: { dataSource: resourceData, ignoreWhitespace: true }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: The property `ignoreWhitespace` will be applicable only when `rowAutoHeight` feature is enabled in the Scheduler.

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Header bar in EJ2 JavaScript Schedule control

The header part of Scheduler can be customized easily with the built-in options available.

Show or Hide header bar

By default, the header bar holds the date and view navigation options, through which the user can switch between the dates and various views. This header bar can be hidden from the UI by setting `false` to the `showHeaderBar` property. Its default value is `true`.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek'],
    showHeaderBar: false,
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing header bar using template

Apart from the default date navigation and view options available on the header bar, you can add custom items into the Scheduler header bar by making use of the [toolbarItems](#) property. To display the default items, it's essential to assign a [name](#) field to each item. The names of the default items are **Previous**, **Next**, **Today**, **DateRangeText**, **NewEvent**, and **Views**. For custom items you can give the name as **Custom** to the [name](#) field. Here, the default items such as previous, next, date range text, and today have been used along with external icon as custom items.

INDEX.TS

```

import { Schedule, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
import { ChangeEventArgs, DropDownList } from '@syncfusion/ej2-dropdowns';
import { Predicate, Query } from '@syncfusion/ej2-data';
Schedule.Inject(Month);
let ownerCollections: Record<string, any>[] = [
  { OwnerText: 'Margaret', OwnerId: 1, Color: '#ea7a57' },
  { OwnerText: 'Robert', OwnerId: 2, Color: '#df5286' },
  { OwnerText: 'Laura', OwnerId: 3, Color: '#865fcf' }
];
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2023, 10, 15),
  views: ['Month'],
  currentView: 'Month',
  resources: [{
    field: 'OwnerId', title: 'Owners',
    name: 'Owners', allowMultiple: true,
    dataSource: ownerCollections,
    textField: 'OwnerText', idField: 'OwnerId', colorField: 'Color',
    query: new Query().where('OwnerId', 'equal', 1)
  }],
  toolbarItems: [{ name: 'Previous', align: 'Left' }, { name: 'Next',
    align: 'Left' }, { name: 'DateRangeText', align: 'Left' },
  {
    type: 'Input', align: 'Center', template: new DropDownList({
      value: 1, showClearButton: false, width: 125,
      fields: { text: 'OwnerText', value: 'OwnerId' },
      dataSource: ownerCollections,
      change: onChange
    })
  }, { name: 'Today', align: 'Right' }],
  eventSettings: { dataSource: scheduleData, query: new
    Query().where('OwnerId', 'equal', 1) }
});
scheduleObj.appendTo('#Schedule');
function onChange(args: ChangeEventArgs): void {
  let resourcePredicate: Predicate;

```

```

let value = args.value;
resourcePredicate = new Predicate('OwnerId', 'equal', value)
scheduleObj.resources[0].query = resourcePredicate ? new
Query().where(resourcePredicate) :
    new Query().where('OwnerId', 'equal', 1);
scheduleObj.eventSettings.query = new Query().where('OwnerId', 'equal',
value);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing header bar using events

Apart from the default date navigation and view options available on the header bar, you can add custom items into the Scheduler header bar by making use of the `actionBegin` event. Here, an employee image is added to the header bar, clicking on which will open the popup showing that person's short profile information.

INDEX.TS

```
import { createElement, compile } from '@syncfusion/ej2-base';
import { Popup } from '@syncfusion/ej2-popups';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { Schedule, Month, ActionEventArgs, ToolbarActionArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Month'],
    currentView: 'Month',
    actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            let userIconItem: ItemModel = {
                align: 'Right', prefixIcon: 'user-icon', text: 'Nancy',
            };
            args.items.push(userIconItem);
        }
    },
    actionComplete: (args: ActionEventArgs) => {
        if (args.requestType === 'toolBarItemRendered') {
            let userIconEle: HTMLElement =
                scheduleObj.element.querySelector('.e-schedule-user-icon') as HTMLElement;
            userIconEle.onclick = () => {
                profilePopup.relateTo = userIconEle;
                profilePopup.dataBind();
                if (profilePopup.element.classList.contains('e-popup-close')) {
                    profilePopup.show();
                } else {
                    profilePopup.hide();
                }
            };
        }
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let userContentEle: HTMLElement = createElement('div', {
    className: 'e-profile-wrapper'
});
scheduleObj.element.parentElement.appendChild(userContentEle);
let userIconEle: HTMLElement = scheduleObj.element.querySelector('.e-schedule-user-icon') as HTMLElement;
```

```

let getDOMString: (data: object) => HTMLCollection = compile('<div
class="profile-container"><div class="profile-image">' +
  '</div><div class="content-wrap"><div class="name">Nancy</div>' +
  '<div class="destination">Product Manager</div><div class="status">'
+
  '<div class="status-icon"></div>Online</div></div></div>');
let output: HTMLCollection = getDOMString({});
let profilePopup: Popup = new Popup(userContentEle, {
  content: output[0] as HTMLElement,
  relateTo: userIconEle,
  position: { X: 'left', Y: 'bottom' },
  collision: { X: 'flip', Y: 'flip' },
  targetType: 'relative',
  viewPortElement: scheduleObj.element,
  width: 185,
  height: 80
});
profilePopup.hide();

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to display the view options within the header bar popup

By default, the header bar holds the view navigation options, through which the user can switch between various views. You can move this view options to the header bar popup by setting `true` to the `enableAdaptiveUI` property.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    enableAdaptiveUI: true,
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Refer [here](#) to know more about adaptive UI in resources scheduler.

Date header customization

The Scheduler UI that displays the date text on all views are considered as the date header cells. You can customize the date header cells of Scheduler either using `dateHeaderTemplate` or `renderCell` event.

Using date header template

The `dateHeaderTemplate` option is used to customize the date header cells of day, week and work-week views.

INDEX.TS

```

import { Schedule, Day, Week, Agenda, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
import { Internationalization } from '@syncfusion/ej2-base';
Schedule.Inject(Day, Week, Agenda, TimelineViews, TimelineMonth);
let instance: Internationalization = new Internationalization();
(window as TemplateFunction).getDateHeaderText = (value: Date) => {
    return instance.formatDate(value, { skeleton: 'Ed' });
};
let getWeather: Function = (value: Date) => {
    switch (value.getDay()) {
        case 0:
            return '<div class="weather-text">25°C</div>';
        case 1:
            return '<div class="weather-text">18°C</div>';
        case 2:
            return '<div class="weather-text">10°C</div>';
        case 3:
            return '<div class="weather-text">16°C</div>';
        case 4:
            return '<div class="weather-text">8°C</div>';
        case 5:

```

```

        return '<div class="weather-text">27°C</div>';
    case 6:
        return '<div class="weather-text">17°C</div>';
    default:
        return null;
    }
};
(window as TemplateFunction).getWeather = getWeather;
interface TemplateFunction extends Window {
    getDateHeaderText?: Function;
    getWeather?: Function;
}
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    dateHeaderTemplate: '<div class="date-
text">${getDateHeaderText(data.date)}</div>${getWeather(data.date)}',
    views: ['Day', 'Week', 'Agenda', 'TimelineWorkWeek', 'TimelineMonth'],
    selectedDate: new Date(2018, 1, 15),
    cssClass: 'schedule-date-header-template',
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using renderCell event

In month view, the date header template is not applicable and therefore the same customization can be added beside the date text in month cells by making use of the `renderCell` event.

INDEX.TS

```

import { Schedule, Month, RenderCellEventArgs } from '@syncfusion/ej2-
schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Month);
let getWeather: Function = (value: Date) => {
    switch (value.getDay()) {
        case 0:
            return '<div class="weather-text">25°C</div>';
        case 1:
            return '<div class="weather-text">18°C</div>';
        case 2:
            return '<div class="weather-text">10°C</div>';
        case 3:
            return '<div class="weather-text">16°C</div>';
        case 4:
            return '<div class="weather-text">8°C</div>';
        case 5:
            return '<div class="weather-text">27°C</div>';
        case 6:
            return '<div class="weather-text">17°C</div>';
        default:
            return null;
    }
};
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Month'],
    renderCell: (args: RenderCellEventArgs) => {
        if (args.elementType === 'monthCells') {
            let ele: Element = document.createElement('div');
            ele.innerHTML = getWeather(args.date);

```

```

        (args.element).appendChild(ele.firstChild);
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing the date range text

The [dateRangeTemplate](#) option allows you to customize the text content of the date range displayed in the scheduler. By default, the date range text is determined by the scheduler view being used. However, you can use the [dateRangeTemplate](#) option to override the default text and specify your own custom text to be displayed.

The [dateRangeTemplate](#) property includes `startDate`, `endDate` and `currentView` options, you can customize the date range text using these available options.

INDEX.TS

```
import { Schedule, Day, Week, Agenda, TimelineViews, TimelineMonth } from
 '@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, Agenda, TimelineViews, TimelineMonth);
(window as any TemplateFunction).getDateRange = (value: Date) =>
 value.toLocaleString('en-us', { month: 'long' }) + ' ' + value.getFullYear();
interface TemplateFunction extends Window {
    getDateRange?: Function;
}
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    dateRangeTemplate: '<div class="date-
text">${getDateRange(data.startDate)}-${getDateRange(data.endDate)}</div>'
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing header indent cells

It is possible to customize the header indent cells using the `headerIndentTemplate` option and change the look and appearance in both the vertical and timeline views. In vertical views, You can customize the header indent cells at the hierarchy level and you can customize the resource header left indent cell in timeline views using the template option.

Example: To customize the header left indent cell to display resources text, refer to the below code example.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, TimelineViews, TimelineMonth } from
'@syncfusion/ej2-schedule';
import { resourceData } from './datasource.ts';
Schedule.Inject( Day, Week, WorkWeek, TimelineViews, TimelineMonth );
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 3, 1),
    headerIndentTemplate: '#indentTemplate',
    views: [
        { option: 'Day' },
        { option: 'Week' },
        { option: 'WorkWeek' },
        { option: 'TimelineWeek' },
        { option: 'TimelineMonth' }
    ],
    group: {
        resources: ['Owners']
    },
    resources: [
        {
            field: 'OwnerId', title: 'Owner',
            name: 'Owners', allowMultiple: true,
            dataSource: [

```

```

        { OwnerText: 'Nancy', Id: 1, OwnerGroupId: 1, OwnerColor:
'#ffaa00' },
        { OwnerText: 'Steven', Id: 2, OwnerGroupId: 2, OwnerColor:
'#f8a398' },
        { OwnerText: 'Michael', Id: 3, OwnerGroupId: 1, OwnerColor:
'#7499e1' }
    ],
    textField: 'OwnerText', idField: 'Id', groupIDField:
'OwnerGroupId', colorField: 'OwnerColor'
    }
    ],
    eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .e-schedule .e-timeline-view .e-resource-left-td {
      vertical-align: bottom;
    }
    .e-schedule .e-timeline-view .e-resource-left-td .e-resource-text,
    .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text
  {
    font-weight: 500;
    padding: 0;
  }
    .e-schedule .e-timeline-view .e-resource-left-td .e-resource-text>div {
      border-right: 1px solid rgba(0, 0, 0, 0.12);
      border-top: 1px solid rgba(0, 0, 0, 0.12);
      flex: 0 0 33.3%;
    }
  }

```

```

        font-weight: 500;
        height: 36px;
        line-height: 34px;
        padding-left: 50px;
    }
    .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div {
        border-right: 1px solid rgba(0, 0, 0, 0.12);
        flex: 0 0 33.3%;
        font-weight: 500;
        height: 36px;
        line-height: 34px;
        padding-left: 50px;
    }
    /* csslint ignore:start */
    .e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-
resource-cells {
        border-bottom-color: rgba(0, 0, 0, 0.12);
    }
    .e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-
resource-cells .e-resource-text {
        font-weight: 500;
    }
    .e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-
header-cells .e-resource-text,
    .e-schedule .e-vertical-view .e-left-indent-wrap table tbody td.e-all-
day-cells .e-resource-text {
        display: none;
    }
    /* csslint ignore:end */
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
    <script id="indentTemplate" type="text/x-template">
        <div class='e-resource-text'>
            <div class="text">Resources</div>
        </div>
    </script>

    <div id="container">
        <div id="Schedule"></div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Timescale in EJ2 JavaScript Schedule control

The time slots are usually the time cells that are displayed on the Day, Week and Work Week views of both the calendar (to the left most position) and timeline views (at the top position). The **timeScale** property allows you to control and set the required time slot duration for the work cells displayed on Scheduler. It includes the following sub-options such as,

- **enable** - When set to **true**, allows the Scheduler to display the appointments accurately against the exact time duration. If set to **false**, all the appointments of a day will be displayed one below the other with no grid lines displayed. Its default value is **true**.
- **interval** – Defines the time duration on which the time axis to be displayed either in 1 hour or 30 minutes interval and so on. It accepts the values in minutes and defaults to 60.
- **slotCount** – Decides the number of slot count to be split for the specified time interval duration. It defaults to 2, thus displaying two slots to represent an hour(each slot depicting 30 minutes duration).

Note: The upper limit for rendering slots within a single day, utilizing the **interval** and **slotCount** properties of the **timeScale**, stands at 1000. This constraint aligns with the maximum **colspan** value permissible for the **table** element, also capped at 1000. This particular restriction is relevant exclusively to the **TimelineDay**, **TimelineWeek** and **TimelineWorkWeek** views.

Setting different time slot duration

The **interval** and **slotCount** properties can be used together on the Scheduler to set different time slot duration which is depicted in the following code example. Here, six time slots together represents an hour.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek'],
    timeScale: {
        enable: true,
        interval: 60,
        slotCount: 6
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script id="majorSlotTemplate" type="text/x-template">
  <div>${majorSlotTemplate(data.date)}</div>
</script><script id="minorSlotTemplate" type="text/x-template">
  <div style="text-align: right; margin-right:
15px">${minorSlotTemplate(data.date)}</div>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing time cells using template

The `timeScale` property also provides template option to allow customization of time slots which are as follows,

- **majorSlotTemplate** - The template option to be applied for major time slots. Here, the template accepts either the string or HTML element as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.
- **minorSlotTemplate** - The template option to be applied for minor time slots. Here, the template accepts either the string or HTML element as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
import { Internationalization } from '@syncfusion/ej2-base';
Schedule.Inject(Day, Week, WorkWeek);
let instance: Internationalization = new Internationalization();
(window as TemplateFunction).majorSlotTemplate = (date: Date) => {
    return instance.formatDate(date, { skeleton: 'hm' });
};
(window as TemplateFunction).minorSlotTemplate = (date: Date) => {
    return instance.formatDate(date, { skeleton: 'ms' }).replace(':00', '');
};
interface TemplateFunction extends Window {
    majorSlotTemplate?: Function;
    minorSlotTemplate?: Function;
}
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek'],
    timeScale: {
        enable: true,
        interval: 60,
        slotCount: 6,
        majorSlotTemplate: '#majorSlotTemplate',
        minorSlotTemplate: '#minorSlotTemplate'
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script id="majorSlotTemplate" type="text/x-template">
  <div>${majorSlotTemplate(data.date)}</div>
</script><script id="minorSlotTemplate" type="text/x-template">
  <div style="text-align: right; margin-right:
15px">${minorSlotTemplate(data.date)}</div>
</script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hide the timescale

The grid lines which indicates the exact time duration can be enabled or disabled on the Scheduler, by setting `true` or `false` to the `enable` option within the `timeScale` property. It's default value is `true`.

INDEX.TS

```

import { Schedule, Day, Week, TimelineViews } from '@syncfusion/ej2-
schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, TimelineViews);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'TimelineWorkWeek'],
  timeScale: {
    enable: false
  },
},

```

```

    eventSettings: { dataSource: scheduleData }
  });
  scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script id="majorSlotTemplate" type="text/x-template">
    <div>${majorSlotTemplate(data.date)}</div>
  </script><script id="minorSlotTemplate" type="text/x-template">
    <div style="text-align: right; margin-right:
15px">${minorSlotTemplate(data.date)}</div>
  </script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Highlighting current date and time

By default, Scheduler indicates current date with a highlighted date header on all views, as well as marks accurately the system's current time on specific views such as Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. To stop highlighting the current time indicator on Scheduler views, set `false` to the `showTimeIndicator` property which defaults to `true`.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, TimelineViews } from
 '@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, WorkWeek, TimelineViews);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Day', 'Week', 'WorkWeek', 'TimelineDay'],
    showTimeIndicator: true
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script id="majorSlotTemplate" type="text/x-template">
    <div>${majorSlotTemplate(data.date)}</div>
  </script><script id="minorSlotTemplate" type="text/x-template">
    <div style="text-align: right; margin-right:
15px">${minorSlotTemplate(data.date)}</div>
  </script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Working days in EJ2 JavaScript Schedule control

The Scheduler can be customized on various aspects as well as it inherits almost all the calendar-specific features such as options,

- To set custom time range display on Scheduler
- To set different working hours
- To set different working days
- To set different first day of week
- To show/hide weekend days
- To show the week number

Set working days

By default, Scheduler considers the week days from Monday to Friday as **Working days** and therefore defaults to [1,2,3,4,5] - where 1 represents Monday, 2 represents Tuesday and so on. The days which are not defined in this working days collection are considered as non-working days. Therefore, when the weekend days are set to hide from Scheduler, all those non-working days too gets hidden from the layout.

The Work week and Timeline Work week views displays exactly the defined working days on Scheduler layout, whereas other views displays all the days and simply differentiates the non-working days on UI with inactive cell color.

The working or business hours depiction on Scheduler are usually valid only on these specified working days.

The following example code depicts how to set the Scheduler to display Monday, Wednesday and Friday as working days of a week.

INDEX.TS

```
import { Schedule, Week, WorkWeek, Month, TimelineViews } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, WorkWeek, Month, TimelineViews);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'WorkWeek',
    views: ['Week', 'WorkWeek', 'Month', 'TimelineWeek',
'TimelineWorkWeek'],
    workDays: [1, 3, 5],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
</body>
</html>
```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Hiding weekend days

The `showWeekend` property is used to either show or hide the weekend days of a week and it is not applicable on Work week view (as non-working days are usually not displayed on work week view). By default, it is set to `true`. The days which are not a part of the working days collection of a Scheduler are usually considered as non-working or weekend days.

Here, the working days are defined as [1, 3, 4, 5] on Scheduler and therefore the remaining days (0, 2, 6 – Sunday, Tuesday and Saturday) are considered as non-working or weekend days and will be hidden from all the views when `showWeekend` property is set to `false`.

INDEX.TS

```

import { Schedule, Day, Week, TimelineMonth } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'TimelineMonth'],
    showWeekend: false,
    workDays: [1, 3, 4, 5],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show week numbers

It is possible to show the week number count of a week in the header bar of the Scheduler by setting true to `showWeekNumber` property. By default, its default value is `false`. In Month view, the week numbers are displayed as a first column.

The `showWeekNumber` property is not applicable on Timeline views, as it has the equivalent `headerRows` property to handle such requirement with additional customizations.

INDEX.TS

```

import { Schedule, Day, Week, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'Month',
    views: ['Day', 'Week', 'Month'],
    showWeekNumber: true,
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```



```

<title>Schedule Typescript Component</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Different options in showing week numbers

By default, week numbers are shown in the Scheduler based on the first day of the year. However, the week numbers can be determined based on the following criteria.

FirstDay – The first week of the year is calculated based on the first day of the year.

FirstFourDayWeek – The first week of the year begins from the first week with four or more days.

FirstFullWeek – The first week of the year begins when meeting the first day of the week (firstDayOfWeek) and the first day of the year.

For more details refer to [this link](#)

INDEX.TS

```
import { Schedule, Day, Week, Month } from '@syncfusion/ej2-schedule';
Schedule.Inject(Day, Week, Month);
let scheduleData: object[] = [
  {
    Id: 1,
    Subject: "Explosion of Betelgeuse Star",
    StartTime: new Date(2020, 1, 15, 9, 30),
    EndTime: new Date(2020, 1, 15, 11, 0)
  },
  {
    Id: 2,
    Subject: "Thule Air Crash Report",
    StartTime: new Date(2020, 1, 12, 12, 0),
    EndTime: new Date(2020, 1, 12, 14, 0)
  },
  {
    Id: 3,
    Subject: "Blue Moon Eclipse",
    StartTime: new Date(2020, 1, 13, 9, 30),
    EndTime: new Date(2020, 1, 13, 11, 0)
  },
  {
    Id: 4,
    Subject: "Meteor Showers in 2018",
    StartTime: new Date(2020, 1, 14, 13, 0),
    EndTime: new Date(2020, 1, 14, 14, 30)
  }
];
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2020, 1, 15),
  currentView: 'Month',
  views: ['Day', 'Week', 'Month'],
  showWeekNumber: true,
  weekRule: 'FirstFourDayWeek',
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: Enable the `showWeekNumber` property to configure the `weekRule` property. Also, the `weekRule` property depends on the value of the `firstDayOfWeek` property.

Set working hours

Working hours indicates the work hour limit within the Scheduler, which is visually highlighted with an active color on work cells. The working hours can be set on Scheduler using the `workHours` property which is of object type and includes the following sub-options,

- `highlight` – enables/disables the highlighting of work hours.
- `start` - sets the start time of the working/business hour of a day.
- `end` - sets the end time limit of the working/business hour of a day.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({

```

```

width: '100%',
height: '550px',
selectedDate: new Date(2018, 1, 15),
views: ['Day', 'Week', 'WorkWeek'],
workHours: {
    highlight: true,
    start: '11:00',
    end: '20:00'
},
eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scheduler displaying custom hours

It is possible to display the event Scheduler layout with specific time durations by hiding the unwanted hours. To do so, set the start and end hour for the Scheduler using the `startHour` and `endHour` properties respectively.

The following code example displays the Scheduler starting from the time range 7.00 AM to 6.00 PM and the remaining hours are hidden on the UI.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek'],
    startHour: '07:00',
    endHour: '18:00',
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Setting start day of the week

By default, Scheduler defaults to **Sunday** as its first day of a week. To change the Scheduler's start day of a week with different day, set the **firstDayOfWeek** property with the values ranging from 0 to 6.

Here, Sunday is always denoted as 0, Monday as 1 and so on.

INDEX.TS

```

import { Schedule, Week, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Week', 'Month'],
    firstDayOfWeek : 3,
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Scroll to specific time and date

You can manually scroll to a specific time on Scheduler by making use of the `scrollTo` method as depicted in the following code example.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { TimePicker, ChangeEventArgs } from '@syncfusion/ej2-calendars';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    width: '100%',
    views: ['Day', 'Week', 'WorkWeek'],
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let scrollToHour: TimePicker = new TimePicker({
    width: 100,

```

```

    value: new Date(2000, 0, 1, 9),
    format: 'HH:mm',
    change: (args: ChangeEventArgs) => {
        scheduleObj.scrollTo(args.text);
    }
});
scrollToHour.appendTo('#ScrollToHour');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table id="property" title="Properties" style="width: 100%">
            <tbody>
                <tr style="height: 50px">
                    <td>
                        <div> Scroll To
                        </div>
                    </td>
                    <td>
                        <div>
                            <input id="ScrollToHour" type="text">
                        </div>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>

```



```

        </tr>
      </tbody>
    </table>
    <div id="Schedule"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to scroll to current time on initial load

There are scenarios where you may need to load the Scheduler displaying the system's current time on the currently visible view port area. In such cases, the Scheduler needs to be scrolled to a specific time based on the system's current time which is depicted in the following code example.

INDEX.TS

```

import { Internationalization } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, TimelineViews } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, TimelineViews);
let intl: Internationalization = new Internationalization();
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  width: '100%',
  views: ['Day', 'Week', 'TimelineWorkWeek'],
  selectedDate: new Date(2018, 1, 15),
  eventSettings: { dataSource: scheduleData },
  created: ():void => {
    scheduleObj.scrollTo(intl.formatDate(new Date(), { skeleton: 'Hm'
  }));
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

See Also

- [To display the current time indicator](#)
- [To set different working hours dynamically](#)
- [To set different working hours for each resources](#)
- [To set different working days for each resources](#)

Cell customization in EJ2 JavaScript Schedule control

The cells of the Scheduler can be easily customized either using the cell template or `renderCell` event.

Setting cell dimensions in all views

The height and width of the Scheduler cells can be customized either to increase or reduce its size through the `cssClass` property, which overrides the default CSS applied on cells.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, ActionEventArgs } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    width: '100%',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    cssClass: 'schedule-cell-dimension',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Check for cell availability

You can check whether the given time range slots are available for event creation or already occupied by other events using the `isSlotAvailable` method. In the following code example, if a specific time slot already contains an appointment, then no more appointments can be added to that cell.

Note: The `isSlotAvailable` is centered around verifying appointments within the present view's date range. Yet, it does not encompass an evaluation of availability for recurrence occurrences that fall beyond this particular date range.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, ActionEventArgs,
EventFieldsMapping } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    actionBegin: (args: ActionEventArgs) => {
        if (args.requestType === 'eventCreate' &&
(<Object[]>args.data).length > 0) {
            let eventData: { [key: string]: Object } = args.data[0] as {
[key: string]: Object };
            let eventField: EventFieldsMapping = scheduleObj.eventFields;
            let startDate: Date = eventData[eventField.startTime] as Date;
            let endDate: Date = eventData[eventField.endTime] as Date;
            args.cancel = !scheduleObj.isSlotAvailable(startDate, endDate);
        }
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing cells in all the views

It is possible to customize the appearance of the cells using both template options and `renderCell` event on all the views.

Using template

The `cellTemplate` option accepts the template string and is used to customize the cell background with specific images or appropriate text on the given date values.

INDEX.TS

```

import { Schedule, Day, Week, Month, TimelineViews } from '@syncfusion/ej2-
schedule';
Schedule.Inject(Day, Week, Month, TimelineViews);
(window as Window).TemplateFunction.getMonthCellText = (date: Date) => {
    if (date.getMonth() === 10 && date.getDate() === 23) {
        return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/birthday.svg" />';
    } else if (date.getMonth() === 11 && date.getDate() === 9) {
        return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/get-together.svg" />';
    } else if (date.getMonth() === 11 && date.getDate() === 13) {
        return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/birthday.svg" />';
    } else if (date.getMonth() === 11 && date.getDate() === 22) {

```

```

        return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/thanksgiving-day.svg"
/>';
    } else if (date.getMonth() === 11 && date.getDate() === 24) {
        return '';
    } else if (date.getMonth() === 11 && date.getDate() === 25) {
        return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/christmas.svg" />';
    } else if (date.getMonth() === 0 && date.getDate() === 1) {
        return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/newyear.svg" />';
    } else if (date.getMonth() === 0 && date.getDate() === 14) {
        return '<img src=
"https://ej2.syncfusion.com/demos/src/schedule/images/birthday.svg" />';
    }
    return '';
};
(window as TemplateFunction).getWorkCellText = (date: Date) => {
    let weekEnds: number[] = [0, 6];
    if (weekEnds.indexOf(date.getDay()) >= 0) {
        return '';
    }
    return '';
};
interface TemplateFunction extends Window {
    getWorkCellText?: Function;
    getMonthCellText?: Function;
}
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Day', 'Week', 'TimelineWeek', 'Month'],
    cssClass: 'schedule-cell-template',
    cellTemplate: '${if(type === "workCells")}<div
class="templatewrap">${getWorkCellText(data.date)}</div>${if}${if(type ===
"monthCells")}<div
class="templatewrap">${getMonthCellText(data.date)}</div>${if}',
    selectedDate: new Date(2017, 11, 16)
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using renderCell event

An alternative to `cellTemplate` is the `renderCell` event, which can also be used to customize the cells with appropriate images or formatted text values.

INDEX.TS

```

import { createElement } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, Month, RenderCellEventArgs } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 14),
    currentView: 'Month',
    views: ['Day', 'Week', 'Month'],
    renderCell: (args: RenderCellEventArgs) => {
        if (args.elementType == 'workCells' || args.elementType ==
 'monthCells') {
            let weekEnds: number[] = [0, 6];

```

```

        if (weekEnds.indexOf((args.date).getDay()) >= 0) {
            let ele: HTMLElement = createElement('div', {
                innerHTML: "<img
src='https://ej2.syncfusion.com/demos/src/schedule/images/newyear.svg' />",
                className: 'templatewrap'
            });
            (args.element).appendChild(ele);
        }
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```



```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can customize cells such as work cells, month cells, all-day cells, header cells, resource header cells using `renderCell` event by checking the `elementType` option within the event. You can check `elementType` with any of the following.

Element type	Description
----- -----	
dateHeader	triggers on header cell rendering.
monthDay	triggers on header cell in month view rendering.
resourceHeader	triggers on resource header cell rendering.
alldayCells	triggers on all day cell rendering.
emptyCells	triggers on empty cell rendering on header bar.
resourceGroupCells	triggers on rendering of work cells for parent resource.
workCells	triggers on work cell rendering.
monthCells	triggers on month cell rendering.
majorSlot	triggers on major time slot cell rendering.
minorSlot	triggers on minor time slot cell rendering.
weekNumberCell	triggers on cell displaying week number.

Customizing cell header in month view

The month header of each date cell in the month view can be customized using the `cellHeaderTemplate` option which accepts the string or `HTMLElement`. The corresponding date can be accessed with the template.

INDEX.TS

```

import { Internationalization } from "@syncfusion/ej2-base";
import { Schedule, Month } from '@syncfusion/ej2-schedule';
Schedule.Inject(Month);
let instance: Internationalization = new Internationalization();
(window as TemplateFunction).getDate = (date: Date) => {
    return instance.formatDate(date, { skeleton: "Ed" });
};
interface TemplateFunction extends Window {
    getDate?: Function;
}
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Month'],
    cssClass: 'schedule-cell-header-template',
    cellHeaderTemplate: '<div class="cell-header-wrap">${getDate(data.date)}</div>'

```

```
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customizing the minimum and maximum date values

Providing the `minDate` and `maxDate` property with some date values, allows the Scheduler to set the minimum and maximum date range. The Scheduler date that lies beyond this minimum and maximum

date range will be in a disabled state so that the date navigation will be blocked beyond the specified date range.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda, Year } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Year);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    views: ['Day', 'Week', 'Month', 'Agenda', 'Year'],
    selectedDate: new Date(2018, 1, 17),
    minDate: new Date(2017, 4, 17),
    maxDate: new Date(2018, 5, 17)
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the `minDate` property value is set to `new Date(1900, 0, 1)` and `maxDate` property value is set to `new Date(2099, 11, 31)`. The user can also set the customized `minDate` and `maxDate` property values.

How to disable multiple cell and row selection in Schedule

By default, the `allowMultiCellSelection` and `allowMultiRowSelection` properties of the Schedule are set to `true`. So, the Schedule allows user to select multiple cells and rows. If the user want to disable this multiple cell and row selection. The user can disable this feature by setting up `false` to these properties.

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

State persistence in EJ2 JavaScript Schedule control

State persistence allowed Scheduler to retain the `currentView`, `selectedDate` and Scroll position values in the `localStorage` for state maintenance even if the browser is refreshed or if you move to the next page within the browser. This action is handled through the `enablePersistence` property which is set to `false` by default. When it is set to `true`, `currentView`, `selectedDate` and Scroll position values of the scheduler component will be retained even after refreshing the page.

Note: Scheduler `id` is essential to set state persistence.

The following sample demonstrates how to set state persistence of the Scheduler component.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '450px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'Month',
    enablePersistence: true,
    views: [ "Day", "Week", "WorkWeek", "Month"],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Exporting in EJ2 JavaScript Schedule control

The Scheduler supports exporting all its appointments both to an Excel or ICS extension file at client-side. It offers different client-side methods to export its appointments in an Excel or iCal format file. Let's look onto the ways on how to implement the exporting functionality in Scheduler.

Excel Exporting

The Scheduler allows you to export all its events into an Excel format file by using the [exportToExcel] client-side method. By default, it exports all the default fields of Scheduler mapped through eventSettings property.

Before you start with excel exporting functionality, you need to import and inject the ExcelExport module from the '@syncfusion/ej2-schedule' package using the Inject method of Scheduler.

INDEX.TS

```
import { Schedule, Week, ExcelExport, ActionEventArgs, ToolbarActionArgs }
  from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, ExcelExport);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2019, 0, 10),
  views: ['Week'],
  eventSettings: { dataSource: scheduleData },
  actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
    if (args.requestType === 'toolbarItemRendering') {
      let exportItem: ItemModel = {
        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
        text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
      };
      args.items.push(exportItem);
    }
  }
});
scheduleObj.appendTo('#Schedule');
function onExportClick(): void {
  scheduleObj.exportToExcel();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting with custom fields

By default, Scheduler exports all the default event fields that are mapped to it through the `eventSettings` property. To limit the number of fields on the exported excel file, it provides an option to export only the custom fields of the event data. To export such custom fields alone, define the required fields through the `ExportOptions` interface and pass it as argument to the `exportToExcel` method as shown in the following example. For example: `['Id', 'Subject', 'StartTime', 'EndTime', 'Location']`.

INDEX.TS

```

import { Schedule, Week, ExcelExport, ExportOptions, ActionEventArgs,
ToolbarActionArgs } from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, ExcelExport);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2019, 0, 10),
    views: ['Week'],
    eventSettings: { dataSource: scheduleData },
    actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            let exportItem: ItemModel = {
                align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',

```

```

                text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
            };
            args.items.push(exportItem);
        }
    }
});
scheduleObj.appendTo('#Schedule');
function onExportClick(): void {
    let exportValues: ExportOptions = { fields: ['Id', 'Subject',
'StartTime', 'EndTime', 'Location'] };
    scheduleObj.exportToExcel(exportValues);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```



```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting individual occurrences of a recurring series

By default, the Scheduler exports recurring events as a single data by exporting only its parent record into the excel file. If you want to export each individual occurrences of a recurring series appointment as separate records in an Excel file, define the `includeOccurrences` option as `true` through the `ExportOptions` interface and pass it as argument to the `exportToExcel` method. By default, the `includeOccurrences` option is set to `false`.

INDEX.TS

```

import { Schedule, Week, ExcelExport, ExportOptions, ActionEventArgs,
ToolbarActionArgs } from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, ExcelExport);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2019, 0, 10),
  views: ['Week'],
  eventSettings: { dataSource: scheduleData },
  actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
    if (args.requestType === 'toolbarItemRendering') {
      let exportItem: ItemModel = {
        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
        text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
      };
      args.items.push(exportItem);
    }
  }
});
scheduleObj.appendTo('#Schedule');
function onExportClick(): void {
  let exportValues: ExportOptions = { includeOccurrences: true };
  scheduleObj.exportToExcel(exportValues);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting custom event data

By default, the whole event collection bound to the Scheduler gets exported as an excel file. To export only specific events of Scheduler or some custom event collection, you need to pass those custom data collection as a parameter to the `exportToExcel` method as shown in this following example, through the `customData` option of `ExportOptions` interface.

By default, the event data are taken from Scheduler `dataSource`.

INDEX.TS

```

import { Schedule, Week, ExcelExport, ExportOptions, ActionEventArgs,
ToolbarActionArgs } from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, ExcelExport);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',

```

```

        selectedDate: new Date(2019, 0, 10),
        views: ['Week'],
        eventSettings: { dataSource: scheduleData },
        actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
            if (args.requestType === 'toolbarItemRendering') {
                let exportItem: ItemModel = {
                    align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
                    text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
                };
                args.items.push(exportItem);
            }
        }
    });
    scheduleObj.appendTo('#Schedule');
    function onExportClick(): void {
        let exportValues: ExportOptions = {
            customData: [{
                Id: 1,
                Subject: 'Explosion of Betelgeuse Star',
                Location: 'Space Centre USA',
                StartTime: new Date(2019, 0, 6, 9, 30),
                EndTime: new Date(2019, 0, 6, 11, 0),
                CategoryColor: '#1aaa55'
            }, {
                Id: 2,
                Subject: 'Thule Air Crash Report',
                Location: 'Newyork City',
                StartTime: new Date(2019, 0, 7, 12, 0),
                EndTime: new Date(2019, 0, 7, 14, 0),
                CategoryColor: '#357cd2'
            }
        ]
    };
    scheduleObj.exportToExcel(exportValues);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing column header with custom fields exporting

Using fields property, we can only export the defined fields into excel without customizing the header. Now we can provide the alternate support to customize the header of custom fields exporting using the fieldsInfo option through the ExportFieldInfo interface and pass it as an argument to the exportToExcel method as shown in the following example.

INDEX.TS

```

import {
    Schedule, ScheduleModel, Week, ExcelExport, ExportOptions,
    ExportFieldInfo, ActionEventArgs
} from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
Schedule.Inject(Week, ExcelExport);
const data: Record<string, any>[] = [
    {
        Id: 1,
        Subject: 'Explosion of Betelgeuse Star',
        StartTime: new Date(2018, 1, 16, 9, 30),
        EndTime: new Date(2018, 1, 16, 11, 0),
        Location: 'Chennai',
        OwnerId: 1
    }, {
        Id: 2,
        Subject: 'Thule Air Crash Report',

```

```

        StartTime: new Date(2018, 1, 12, 12, 0),
        EndTime: new Date(2018, 1, 12, 14, 0),
        Location: 'Mumbai',
        OwnerId: 2
    }, {
        Id: 3,
        Subject: 'Blue Moon Eclipse',
        StartTime: new Date(2018, 1, 13, 9, 30),
        EndTime: new Date(2018, 1, 13, 11, 0),
        Location: 'Mumbai',
        OwnerId: 3
    }, {
        Id: 4,
        Subject: 'Meteor Showers in 2018',
        StartTime: new Date(2018, 1, 14, 13, 0),
        EndTime: new Date(2018, 1, 14, 14, 30),
        Location: 'Bangalore',
        OwnerId: 1
    }, {
        Id: 5,
        Subject: 'Milky Way as Melting pot',
        StartTime: new Date(2018, 1, 15, 12, 0),
        EndTime: new Date(2018, 1, 15, 14, 0),
        Location: 'Chennai',
        OwnerId: 2
    }
];
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: 500,
    selectedDate: new Date(2018, 1, 15),
    views: [
        { option: 'Week' }
    ],
    resources: [
        {
            field: 'OwnerId', title: 'Owner',
            name: 'Owners',
            dataSource: [
                { OwnerText: 'Nancy', Id: 1, OwnerColor: '#ffaa00' },
                { OwnerText: 'Steven', Id: 2, OwnerColor: '#f8a398' },
                { OwnerText: 'Michael', Id: 3, OwnerColor: '#7499e1' }
            ],
            textField: 'OwnerText', idField: 'Id', colorField: 'OwnerColor'
        }
    ],
    eventSettings: { dataSource: data },
    actionBegin: (args: ActionEventArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            const exportItem: ItemModel = {
                align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
                text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
            };
            args.items.push(exportItem);
        }
    }
});

```

```

    }
  });
  scheduleObj.appendTo('#Schedule');
  function onExportClick(): void {
    const customFields: ExportFieldInfo[] = [
      { name: 'Subject', text: 'Summary' },
      { name: 'StartTime', text: 'First Date' },
      { name: 'EndTime', text: 'Last Date' },
      { name: 'Location', text: 'Place' },
      { name: 'OwnerId', text: 'Owners' }
    ];
    const exportValues: ExportOptions = { fieldsInfo: customFields };
    scheduleObj.exportToExcel(exportValues);
  }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Export with custom file name

By default, the Scheduler allows you to download the exported Excel file with a name `Schedule.xlsx`. It also provides an option to export the excel file with a custom file name, by defining the desired `fileName` through the `ExportOptions` interface and passing it as an argument to the `exportToExcel` method.

INDEX.TS

```

import { Schedule, Week, ExcelExport, ExportOptions, ActionEventArgs,
ToolbarActionArgs } from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, ExcelExport);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2019, 0, 10),
    views: ['Week'],
    eventSettings: { dataSource: scheduleData },
    actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            let exportItem: ItemModel = {
                align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
                text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
            };
            args.items.push(exportItem);
        }
    }
});
scheduleObj.appendTo('#Schedule');
function onExportClick(): void {
    let exportValues: ExportOptions = { fileName: "SchedulerData" };
    scheduleObj.exportToExcel(exportValues);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Excel file formats

By default, the Scheduler exports event data to an excel file in the .xlsx format. You can also export the Scheduler data in either of the file type such as .xlsx or csv formats, by defining the exportType option as either csv or xlsx through the ExportOptions interface. By default, the exportType is set to xlsx.

INDEX.TS

```

import { Schedule, Week, ExcelExport, ExportOptions, ActionEventArgs,
ToolbarActionArgs } from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Week, ExcelExport);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2019, 0, 10),
    views: ['Week'],
    eventSettings: { dataSource: scheduleData },

```



```

    actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            let exportItem: ItemModel = {
                align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
                text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
            };
            args.items.push(exportItem);
        }
    }
});
scheduleObj.appendTo('#Schedule');
function onExportClick(): void {
    let exportValues: ExportOptions = { exportType: "csv" };
    scheduleObj.exportToExcel(exportValues);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Custom separator in CSV

The Scheduler exports the event data to CSV format with `,` as separator. You can change separator by setting [separator](#) property in [ExportOptions](#).

INDEX.TS

```

import { Schedule, Week, ExcelExport, ExportOptions, ExportFieldInfo,
ActionEventArgs, ToolbarActionArgs } from '@syncfusion/ej2-schedule';
import { ItemModel } from '@syncfusion/ej2-navigations';
Schedule.Inject(Week, ExcelExport);
const data: object[] = [
    {
        Id: 1,
        Subject: 'Explosion of Betelgeuse Star',
        StartTime: new Date(2023, 0, 8, 9, 30),
        EndTime: new Date(2023, 0, 8, 11, 0),
        Location: 'Chennai',
        OwnerId: 1
    }, {
        Id: 2,
        Subject: 'Thule Air Crash Report',
        StartTime: new Date(2023, 0, 10, 12, 0),
        EndTime: new Date(2023, 0, 10, 14, 0),
        Location: 'Mumbai',
        OwnerId: 2
    }, {
        Id: 3,
        Subject: 'Blue Moon Eclipse',
        StartTime: new Date(2023, 0, 13, 9, 30),
        EndTime: new Date(2023, 0, 13, 11, 0),
        Location: 'Mumbai',
        OwnerId: 3
    }
];
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '550px',
    selectedDate: new Date(2023, 0, 10),
    views: ['Week'],
    eventSettings: { dataSource: data },
    actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            let exportItem: ItemModel = {
                align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',

```

```

                text: 'CSV-Export', cssClass: 'e-excel-export', click:
onExportClick
                };
                args.items.push(exportItem);
            }
        }
    });
    scheduleObj.appendTo('#Schedule');
    function onExportClick(): void {
        let exportValues: ExportOptions = {exportType: 'csv', separator: ','};
        scheduleObj.exportToExcel(exportValues);
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting calendar events as ICS file

You can export the Scheduler events to a calendar (.ics) file format, and open it on any of the other default calendars such as Google or Outlook. To export the events of Scheduler to an ICS file, you need to first import the `ICalendarExport` module from `@syncfusion/ej2-schedule` package and then inject it using the `Schedule.Inject(ICalendarExport)` method.

The following code example shows how the Scheduler events are exported to a calendar (.ics) file by making use of the `exportToCalendar` public method.

INDEX.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, ICalendarExport } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, ICalendarExport);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '520px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let buttonObj: Button = new Button();
buttonObj.appendTo('#ics-export');
buttonObj.element.onclick = () => scheduleObj.exportToICalendar();

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="ics-export" type="button" value="Export">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Exporting calendar with custom file name

By default, the calendar is exported with a file name **Calendar.ics**. To change this file name on export, pass the custom string value as **fileName** to the method argument so as to get the file downloaded with this provided name.

The following example downloads the iCal file with a name **ScheduleEvents.ics**.

INDEX.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, ICalendarExport } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, ICalendarExport);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '520px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let buttonObj: Button = new Button();
buttonObj.appendTo('#ics-export');
buttonObj.element.onclick = () =>
    scheduleObj.exportToICalendar('ScheduleEvents');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input id="ics-export" type="button" value="Export">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Import events from other calendars

The events from external calendars (ICS files) can be imported into Scheduler by using the `importICalendar` method. This method accepts the `blob object` of an .ics file to be imported, as a mandatory argument.

To import an ICS file containing events into Scheduler, you need to first import the `ICalendarImport` module from `@syncfusion/ej2-schedule` package and then inject it using the `Schedule.Inject(ICalendarImport)` method.

The following example shows how to import an ICS file into Scheduler, using the `importICalendar` method.

INDEX.TS

```
import { Uploader, SelectedEventArgs } from '@syncfusion/ej2-inputs';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, ICalendarImport } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, ICalendarImport);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '520px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month', 'Agenda'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
// Initialize Uploader component for import
let uploadObj: Uploader = new Uploader({
    allowedExtensions: '.ics',
    cssClass: 'calendar-import',
    buttons: { browse: 'Choose file' },
    multiple: false,
    showFileList: false,
    selected: (args: SelectedEventArgs) =>
scheduleObj.importICalendar((<HTMLInputElement>args.event.target).files[0])
});
uploadObj.appendTo('#ics-import');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input id="ics-import" type="file">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to print the Scheduler element

The Scheduler allows you to print the Scheduler element by using the `print` client-side method. The `print` method works in two ways. You can find it below.

- Using print method without options.
- Using a print method with options.

To print the Schedule, you need to import the `Print` module from the `@syncfusion/ej2-schedule` package and then inject it using the `Schedule.Inject(Print)` method.

Using print method without options

You can print the Schedule element with the current view by using the `print` method without passing the options. The following example shows how to print the Scheduler using the `print` method without passing options.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, ActionEventArgs,
ToolbarActionArgs, Print } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Print);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '520px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData },
    actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            let printItem: ItemModel = {

```



```

        align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-print',
        text: 'Print', cssClass: 'e-schedule-print', click:
onPrintIconClick
    };
    args.items.push(printItem);
}
}
});
scheduleObj.appendTo('#Schedule');
function onPrintIconClick(): void {
    scheduleObj.print();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Using a print method with options

You can print the Schedule element based on your needs using the `print` method by passing the print options used in this example with its values. The following example shows how to print the Scheduler using the `print` method by passing the options.

INDEX.TS

```

import { extend } from '@syncfusion/ej2-base';
import { ItemModel } from '@syncfusion/ej2-navigations';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, ActionEventArgs,
ToolbarActionArgs, Print, ScheduleModel } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Print);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '520px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData },
    actionBegin: (args: ActionEventArgs & ToolbarActionArgs) => {
        if (args.requestType === 'toolbarItemRendering') {
            let printItem: ItemModel = {
                align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-print',
                text: 'Print', cssClass: 'e-schedule-print', click:
onPrintIconClick
            };
            args.items.push(printItem);
        }
    }
});
scheduleObj.appendTo('#Schedule');
function onPrintIconClick(): void {
    let printModel: ScheduleModel = {
        agendaDaysCount: 14,
        cssClass: 'e-print-schedule',
        currentView: scheduleObj.currentView,
        dateFormat: 'dd-MMM-yyyy',
        enableRtl: false,
        endHour: '18:00',
        firstDayOfWeek: 1,
        firstMonthOfYear: 0,
        group: {},
        height: 'auto',
        locale: scheduleObj.locale,
        maxDate: scheduleObj.selectedDate,
        minDate: scheduleObj.getCurrentViewDates()[0],
        readonly: true,
        resources: [],
        rowAutoHeight: false,
        selectedDate: new Date(),
        showHeaderBar: false,

```

```

        showTimeIndicator: false,
        showWeekNumber: false,
        showWeekend: false,
        startHour: '06:00',
        timeFormat: 'HH',
        timeScale: { enable: true },
        width: 'auto',
        workDays: [1, 2, 3, 4, 5],
        workHours: { highlight: true, start: '10:00', end: '20:00' }
    };
    scheduleObj.print(printModel);
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Context menu in EJ2 JavaScript Schedule control

You can display context menu on work cells and appointments of Scheduler by making use of the **ContextMenu** control manually from the application end. In the following code example, context menu control is being added from sample end and set its target as **Scheduler**.

On Scheduler cells, you can display the menu items such as **New Event**, **New Recurring Event** and **Today** option. For appointments, you can display its related options such as **Edit Event** and **Delete Event**. The default event window can be opened for appointment creation and editing using the **openEditor** method of Scheduler.

The deletion of appointments can be done by using the **deleteEvent** public method. Also, the **selectedDate** property can be used to navigate between different dates.

You can also display custom menu options on Scheduler cells and appointments. Context menu will open on tap-hold in responsive mode.

INDEX.TS

```

import { closest, isNullOrUndefined, removeClass, remove, extend } from
 '@syncfusion/ej2-base';
import { Query, DataManager } from '@syncfusion/ej2-data';
import {
  Schedule, Day, Week, WorkWeek, Month, Agenda, CellClickEventArgs
} from '@syncfusion/ej2-schedule';
import {
  ContextMenu, MenuItemModel, BeforeOpenCloseMenuEventArgs, MenuEventArgs
} from '@syncfusion/ej2-navigations';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  width: '100%',
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let selectedTarget: Element;
let menuObj: ContextMenu;
let menuItems: MenuItemModel[] = [
  {
    text: 'New Event',
    iconCss: 'e-icons new',
    id: 'Add'
  }, {
    text: 'New Recurring Event',
    iconCss: 'e-icons recurrence',

```

```

        id: 'AddRecurrence'
    }, {
        text: 'Today',
        iconCss: 'e-icons today',
        id: 'Today'
    }, {
        text: 'Edit Event',
        iconCss: 'e-icons edit',
        id: 'Save'
    }, {
        text: 'Edit Event',
        id: 'EditRecurrenceEvent',
        iconCss: 'e-icons edit',
        items: [{
            text: 'Edit Occurrence',
            id: 'EditOccurrence'
        }, {
            text: 'Edit Series',
            id: 'EditSeries'
        }]
    }, {
        text: 'Delete Event',
        iconCss: 'e-icons delete',
        id: 'Delete'
    }, {
        text: 'Delete Event',
        id: 'DeleteRecurrenceEvent',
        iconCss: 'e-icons delete',
        items: [{
            text: 'Delete Occurrence',
            id: 'DeleteOccurrence'
        }, {
            text: 'Delete Series',
            id: 'DeleteSeries'
        }]
    }
];
menuObj = new ContextMenu({
    target: '.e-schedule',
    items: menuItems,
    beforeOpen: onContextMenuBeforeOpen,
    select: onMenuItemSelect,
    cssClass: 'schedule-context-menu'
});
menuObj.appendTo('#ContextMenu');
function onContextMenuBeforeOpen(args: BeforeOpenCloseMenuEventArgs): void {
    let newEventElement: HTMLElement = document.querySelector('.e-new-event') as HTMLElement;
    if (newEventElement) {
        remove(newEventElement);
        removeClass([document.querySelector('.e-selected-cell')], 'e-selected-cell');
    }
    scheduleObj.closeQuickInfoPopup();
    let targetElement: HTMLElement = <HTMLElement>args.event.target;
    if (closest(targetElement, '.e-contextmenu')) {
        return;
    }
}

```

```

    }
    selectedTarget = closest(targetElement, '.e-appointment,.e-work-cells,'
+
    '.e-vertical-view .e-date-header-wrap .e-all-day-cells,.e-vertical-
view .e-date-header-wrap .e-header-cells');
    if (isNullOrUndefined(selectedTarget)) {
        args.cancel = true;
        return;
    }
    if (selectedTarget.classList.contains('e-appointment')) {
        let eventObj: { [key: string]: Object } = <{ [key: string]: Object
}>scheduleObj.getEventDetails(selectedTarget);
        if (eventObj.RecurrenceRule) {
            menuObj.showItems(['EditRecurrenceEvent',
'DeleteRecurrenceEvent'], true);
            menuObj.hideItems(['Add', 'AddRecurrence', 'Today', 'Save',
'Delete'], true);
        } else {
            menuObj.showItems(['Save', 'Delete'], true);
            menuObj.hideItems(['Add', 'AddRecurrence', 'Today',
'EditRecurrenceEvent', 'DeleteRecurrenceEvent'], true);
        }
        return;
    }
    menuObj.hideItems(['Save', 'Delete', 'EditRecurrenceEvent',
'DeleteRecurrenceEvent'], true);
    menuObj.showItems(['Add', 'AddRecurrence', 'Today'], true);
}
function onMenuItemSelect(args: MenuEventArgs): void {
    let selectedMenuItem: string = args.item.id;
    let eventObj: { [key: string]: Object };
    if (selectedTarget && selectedTarget.classList.contains('e-
appointment')) {
        eventObj = <{ [key: string]: Object
}>scheduleObj.getEventDetails(selectedTarget);
    }
    switch (selectedMenuItem) {
        case 'Today':
            scheduleObj.selectedDate = new Date();
            break;
        case 'Add':
        case 'AddRecurrence':
            let selectedCells: Element[] =
scheduleObj.getSelectedElements();
            let activeCellsData: CellClickEventArgs =
scheduleObj.getCellDetails(selectedCells.length > 0 ? selectedCells :
selectedTarget);
            if (selectedMenuItem === 'Add') {
                scheduleObj.openEditor(activeCellsData, 'Add');
            } else {
                scheduleObj.openEditor(activeCellsData, 'Add', null, 1);
            }
            break;
        case 'Save':
        case 'EditOccurrence':
        case 'EditSeries':
            if (selectedMenuItem === 'EditSeries') {

```

```

        eventObj = <{ [key: string]: Object }>new
DataManager(scheduleObj.eventsData).executeLocal(new Query().
    where(scheduleObj.eventFields.id, 'equal',
eventObj[scheduleObj.eventFields.recurrenceID] as string | number))[0];
    }
    scheduleObj.openEditor(eventObj, selectedMenuItem);
    break;
case 'Delete':
    scheduleObj.deleteEvent(eventObj);
    break;
case 'DeleteOccurrence':
case 'DeleteSeries':
    scheduleObj.deleteEvent(eventObj, selectedMenuItem);
    break;
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="content-wrapper">
            <div id="Schedule">
                </div>
            </div>
        </div>
    </body>

```

```

        <ul id="ContextMenu"></ul>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Dimensions in EJ2 JavaScript Schedule control

The Scheduler dimensions refers to both height and width of the entire layout and it accepts 3 types of values.

- auto
- pixel
- percentage

Auto Height and Width

When height and width of the Scheduler are set to **auto**, it will try as hard as possible to keep an element the same width as its parent container. In other words, the parent container that holds Scheduler, its width/height will be the sum of its children. By default, Scheduler is assigned with **auto** values for both height and width properties.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
 '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: 'auto', height: 'auto',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Height and Width in pixel

The Scheduler height and width will be rendered exactly as per the given pixel values. It accepts both string and number values.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '650px', height: '550px',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Height and Width in percentage

When height and width of the Scheduler are given as percentage, it will make the Scheduler as wide as the parent container.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';

```

```
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%', height: '100%',
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

See Also

- [How to Change Scheduler Cell Dimensions](#)

Recurrence editor in EJ2 JavaScript Schedule control

The Recurrence editor is integrated into Scheduler editor window by default, to process the recurrence rule generation for events. Apart from this, it can also be used as an individual component referring from the Scheduler repository to work with the recurrence related processes.

All the valid recurrence rule string mentioned in the [iCalendar](#) specifications are applicable to use with the recurrence editor.

Customizing the repeat type option in editor

By default, there are 5 types of repeat options available in recurrence editor such as,

- Never
- Daily
- Weekly
- Monthly
- Yearly

It is possible to customize the recurrence editor to display only the specific repeat options such as **Daily** and **Weekly** options alone by setting the appropriate **frequencies** option.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month, Agenda, RecurrenceEditor,
  PopupOpenEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
  width: '100%', height: '550px',
  selectedDate: new Date(2018, 1, 15),
  eventSettings: { dataSource: scheduleData },
  popupOpen: (args: PopupOpenEventArgs): void => {
    if (args.type == 'Editor') {
      scheduleObj.eventWindow.recurrenceEditor.frequencies = ['none',
        'daily', 'weekly'];
    }
  }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The other properties available in recurrence editor are tabulated below,

Properties	Type	Description
----- ----- -----		
firstDayOfWeek	number	Sets the first day of the week on recurrence editor.
startDate	Date	Sets the start date from which date the recurrence event starts.
dateFormat	string	Sets the specific date format on recurrence editor.
locale	string	Sets the locale to be applied on recurrence editor.
cssClass	string	Allows styling to be applied on recurrence editor with custom class names.
enableRtl	boolean	Allows recurrence editor to render in RTL mode.

- | minDate | Date | Sets the minimum date on recurrence editor. |
- | maxDate | Date | Sets the maximum date on recurrence editor. |
- | value | string | Sets the recurrence rule value on recurrence editor. |
- | selectedType | number | Sets the specific repeat type on the recurrence editor. |

Customizing the End Type Option in Editor

By default, there are 3 types of end options available in the recurrence editor such as:

- Never
- Until
- Count

It is possible to customize the recurrence editor to display only the specific end options, such as the **Until** and **Count** options alone, by setting the appropriate [endTypes](#) option.

INDEX.TS

```
import { RecurrenceEditor } from '@syncfusion/ej2-schedule';
let recObject: RecurrenceEditor = new RecurrenceEditor({
    frequencies: ['daily', 'weekly'],
    endTypes: ['until', 'count']
});
recObject.appendTo('#RecurrenceEditor');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id='loader'>LOADING....</div>
  <div id="container">
    <div id="RecurrenceEditor"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Accessing the recurrence rule string

The recurrence rule is usually generated based on the options selected from the recurrence editor and also it follows the [iCalendar](#) specifications. The generated recurrence rule string is a valid one to be used with the Scheduler event's recurrence rule field.

There is a `change` event available in recurrence editor, that triggers on every time the fields of recurrence editor tends to change. Within this event argument, you can access the generated recurrence value through the `value` option as shown in the following code example.

INDEX.TS

```

import { RecurrenceEditor, RecurrenceEditorChangeEventArgs } from
'@syncfusion/ej2-schedule';
let outputElement: HTMLElement = <HTMLElement>document.querySelector('#rule-
output');
let recObject: RecurrenceEditor = new RecurrenceEditor({
  change: (args: RecurrenceEditorChangeEventArgs) => {
    if(args.value == "") {
      outputElement.innerText = 'Select Rule';
    } else {
      outputElement.innerText = args.value;
    }
  }
});
recObject.appendTo('#RecurrenceEditor');
outputElement.innerText = 'Select Rule';

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .recurrence-editor-wrap {
        margin: 0 25%;
    }
    .rule-output-container {
        height: auto;
        border: 1px solid #969696;
    }
    #rule-output {
        padding: 8px 4px;
        text-align: center;
        min-height: 20px;
        overflow: hidden;
        overflow-wrap: break-word;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="padding-bottom:15px;">
            <label id="rule-label">Rule Output</label>
            <div class="rule-output-container">
                <div id="rule-output"></div>
            </div>
        </div>
        <div id="RecurrenceEditor"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>

```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Set specific value on recurrence editor

It is possible to display the recurrence editor with specific options loaded initially, based on the rule string that we provide. The fields of recurrence editor will change its values accordingly, when we provide a particular rule through the `setRecurrenceRule` method.

INDEX.TS

```
import { RecurrenceEditor, RecurrenceEditorChangeEventArgs } from
 '@syncfusion/ej2-schedule';
let outputElement: HTMLElement = <HTMLElement>document.querySelector('#rule-
output');
let recObject: RecurrenceEditor = new RecurrenceEditor({
  change: (args: RecurrenceEditorChangeEventArgs) => {
    outputElement.innerText = args.value;
  }
});
recObject.appendTo('#RecurrenceEditor');
recObject.setRecurrenceRule('FREQ=DAILY;INTERVAL=2;COUNT=8');
outputElement.innerText = recObject.value as string;
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .recurrence-editor-wrap {
      margin: 0 25%;
    }
    .rule-output-container {
```

```

        height: auto;
        border: 1px solid #969696;
    }
    #rule-output {
        padding: 8px 4px;
        text-align: center;
        min-height: 20px;
        overflow: hidden;
        overflow-wrap: break-word;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="padding-bottom:15px;">
            <label id="rule-label">Rule Output</label>
            <div class="rule-output-container">
                <div id="rule-output"></div>
            </div>
        </div>
        <div id="RecurrenceEditor"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Recurrence date generation

You can parse the `recurrenceRule` of an event to generate the date instances on which that particular event is going to occur, using the `getRecurrenceDates` method. It generates the dates based on the `recurrenceRule` that we provide. The parameters to be provided for `getRecurrenceDates` method are as follows.

Field name	Type	Description
----- ----- -----		
<code>startDate</code>	Date	Appointment start date.
<code>rule</code>	String	Recurrence rule present in an event object.
<code>excludeDate</code>	String	Date collection (in ISO format) to be excluded. It is optional .
<code>maximumCount</code>	Number	Number of date count to be generated. It is optional .
<code>viewDate</code>	Date	Current view range's first date. It is optional .

INDEX.TS

```

import { createElement } from '@syncfusion/ej2-base';
import { RecurrenceEditor, RecurrenceEditorChangeEventArgs } from
 '@syncfusion/ej2-schedule';
let outputElement: HTMLElement = <HTMLElement>document.querySelector('#rule-
output');
let labelElement: HTMLElement = <HTMLElement>document.querySelector('#rule-
label');
let recObject: RecurrenceEditor = new RecurrenceEditor({
    change: OnRecurrenceChange,
});
recObject.appendTo('#RecurrenceEditor');
let ruleString: string = 'FREQ=DAILY;INTERVAL=1';
recObject.setRecurrenceRule(ruleString);
labelElement.innerText = 'Date Collections';
outputElement.innerHTML = '';
let ruleStringDates: number[] = recObject.getRecurrenceDates(
    new Date(),
    ruleString
);
for (let index: number = 0; index < ruleStringDates.length; index++) {
    outputElement.appendChild(
        createElement('div', {
            innerHTML: new Date(ruleStringDates[index]).toString(),
        })
    );
}
function OnRecurrenceChange(args: RecurrenceEditorChangeEventArgs): void {
    if (args.value == '') {
        outputElement.innerText = 'Select Rule';
    } else {
        outputElement.innerHTML = '';
        let dates: number[] = recObject.getRecurrenceDates(new Date(),
args.value);
        for (let index: number = 0; index < dates.length; index++) {
            outputElement.appendChild(
                createElement('div', { innerHTML: new Date(dates[index]).toString()
            })
        );
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .recurrence-editor-wrap {
        margin: 0 25%;
    }
    .rule-output-container {
        height: auto;
        border: 1px solid #969696;
    }
    #rule-output {
        padding: 8px 4px;
        text-align: center;
        min-height: 20px;
        overflow: hidden;
        overflow-wrap: break-word;
    }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div style="padding-bottom:15px;">
            <label id="rule-label">Rule Output</label>
            <div class="rule-output-container">
                <div id="rule-output"></div>
            </div>
        </div>
        <div id="RecurrenceEditor"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Above example will generate two dates January 7, 2018 & January 9 2018 by excluding the in between dates January 8 2018 & January 10 2018, since those dates were given in the exclusion list. Generated dates can then be utilised to create appointments.

Recurrence date generation in server-side

It is also possible to generate recurrence date instances from server-side by manually referring the **RecurrenceHelper** class, which is specifically written and referred from application end to handle this date generation process.

Refer [here](#) for the step by step procedure to achieve date generation in server-side.

Restrict date generation with specific count

In case, if the rule is given in "NEVER ENDS" category, then you can mention the maximum count when you actually want to stop the date generation starting from the provided start date. To do so, provide the appropriate **maximumCount** value within the **getRecurrenceDates** method as shown in the following code example.

INDEX.TS

```
import { createElement } from '@syncfusion/ej2-base';
import { RecurrenceEditor } from '@syncfusion/ej2-schedule';
import { NumericTextBox, ChangeEventArgs } from '@syncfusion/ej2-inputs';
let numericValue: number = 10;
let numeric: NumericTextBox = new NumericTextBox({
    value: numericValue,
    change: numericFocus,
});
numeric.appendTo('#numeric');
let outputElement: HTMLElement = <HTMLElement>document.querySelector('#rule-output');
let labelElement: HTMLElement = <HTMLElement>document.querySelector('#rule-label');
let ruleString: string = 'FREQ=DAILY;INTERVAL=1; COUNT=30';
let recObject: RecurrenceEditor = new RecurrenceEditor();
let dates: number[] =
recObject.getRecurrenceDates(new Date(2018, 0, 7, 10, 0), ruleString,
'20180108T114224Z,20180110T114224Z', numericValue, new Date(2018, 0, 7));
labelElement.innerText = 'Date Collections';
outputElement.innerHTML = '';
for (let index: number = 0; index < dates.length; index++) {
    outputElement.appendChild(createElement('div', { innerHTML: new
Date(dates[index]).toString() }));
}
function numericFocus(args: ChangeEventArgs): void {
    outputElement.innerHTML = '';
    let dates: number[] = recObject.getRecurrenceDates(
        new Date(2018, 0, 7, 10, 0),
        ruleString,
        '20180108T114224Z,20180110T114224Z',
        args.value,
        new Date(2018, 0, 7)
    );
    for (let index: number = 0; index < dates.length; index++) {
        outputElement.appendChild(
```

```

        createElement('div', { innerHTML: new Date(dates[index]).toString() })
    );
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <style>
    .recurrence-editor-wrap {
      margin: 0 25%;
    }
    .rule-output-container {
      height: auto;
      border: 1px solid #969696;
    }
    #rule-output {
      padding: 8px 4px;
      text-align: center;
      min-height: 20px;
      overflow: hidden;
      overflow-wrap: break-word;
    }
  </style>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">

```

```

        <div style="padding-bottom:15px;">
            <label id="rule-label">Rule Output</label>
            <div class="rule-output-container">
                <div id="rule-output"></div>
            </div>
        </div>
        <div class="wrap">
            <input id="numeric" type="text" />
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Localization in EJ2 JavaScript Schedule control

The Scheduler integrates different date-time formats and cultures, which allows it to function globally, thus meeting the diverse needs of different regions.

You can adapt the Scheduler to various languages by parsing and formatting the date or number ([Internationalization](#)), adding culture specific customization and translation to the text ([Localization](#)).

Globalization

The Internationalization library provides support for formatting and parsing the number, date, and time by using the official [Unicode CLDR](#) JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data.

By default, Scheduler is set to follow the English culture ('en-US'). If you want to go with different culture other than English, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this [link](#).

```
npm install cldr-data --save
```

Once the package is installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Now import the installed CLDR JSON data into the `app.ts` file. To import JSON data, you need to install the JSON plugin loader. Here, we have used the SystemJS JSON plugin loader.

```
npm install systemjs-plugin-json --save-dev
```

- Once installed, configure the `system.config.js` configuration settings as shown in the following code to map the `systemjs-plugin-json` loader.

```
`ts
System.config({
  paths: {
    'syncfusion:': 'npm:@syncfusion/'
  },
  map: {
    app: 'app',
    //Syncfusion packages mapping
    "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
    "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
    "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
    "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
    "@syncfusion/ej2-dropdowns": "syncfusion:ej2-dropdowns/dist/ej2-dropdowns.umd.min.js",
    "@syncfusion/ej2-compression": "syncfusion:ej2-compression/dist/ej2-compression.umd.min.js",
    "@syncfusion/ej2-file-utils": "syncfusion:ej2-file-utils/dist/ej2-file-utils.umd.min.js",
    "@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
    "@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
    "@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",
    "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
    "@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
    "@syncfusion/ej2-excel-export": "syncfusion:ej2-excel-export/dist/ej2-excel-export.umd.min.js",
    "@syncfusion/ej2-schedule": "syncfusion:ej2-schedule/dist/ej2-schedule.umd.min.js",
    "cldr-data": 'npm:cldr-data',
    "plugin-json": "npm:systemjs-plugin-json/json.js"
  },
  meta: {
    '*.json': { loader: 'plugin-json' }
  },
}
```



```
packages: {
  'app': { main: 'app', defaultExtension: 'js' },
  'cldr-data': { main: 'index.js', defaultExtension: 'js' }
}
});
System.import('app');
`
```

- Now import the required cultures from the installed location to `app.ts` file as given in the following code example.

```
`ts
//import the loadCldr from ej2-base
import { loadCldr } from '@syncfusion/ej2-base';
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/fr-CH/ca-gregorian.json'),
  require('cldr-data/main/fr-CH/numbers.json'),
  require('cldr-data/main/fr-CH/timeZoneNames.json')
);
`
```

- Set the culture to Scheduler by using the `locale` property.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
import { loadCldr } from '@syncfusion/ej2-base';
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
Schedule.Inject(Day, Week, WorkWeek, Month);
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames);
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  locale: 'fr-CH',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'WorkWeek', 'Month'],
  eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Localizing the static Scheduler text

[Localization](#) library allows to display all the static text, date content, and time mode of the Scheduler following the localized language. To achieve this, set the **locale** property of Scheduler, as well as define the translation text of static words of Scheduler through the **load** method.

For example, the following code example lets you to define the French translation words for all the static words used in Scheduler.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { L10n, loadCldr } from '@syncfusion/ej2-base';
import { scheduleData } from './datasource.ts';
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as localeTexts from './locale.json';
Schedule.Inject(Day, Week, WorkWeek, Month);
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load(localeTexts);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    locale: 'fr-CH',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
```

```
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The localized words for static text used in Scheduler and Recurrence Editor can be referred from the following code.

```
`ts
L10n.load({
"en": {
"schedule": {
"day": "Day",
"week": "Week",
"workWeek": "Work Week",
"month": "Month",
"agenda": "Agenda",
"weekAgenda": "Week Agenda",
"workWeekAgenda": "Work Week Agenda",
"monthAgenda": "Month Agenda",
"today": "Today",
"noEvents": "No events",
"emptyContainer": "There are no events scheduled on this day.",
"allDay": "All day",
"start": "Start",
"end": "End",
"more": "more",
"close": "Close",
"cancel": "Cancel",
"noTitle": "(No Title)",
"delete": "Delete",
```

```
"deleteEvent": "Delete Event",
"deleteMultipleEvent": "Delete Multiple Events",
"selectedItems": "Items selected",
"deleteSeries": "Delete Series",
"edit": "Edit",
"editSeries": "Edit Series",
"editEvent": "Edit Event",
"createEvent": "Create",
"subject": "Subject",
"addTitle": "Add title",
"moreDetails": "More Details",
"save": "Save",
"editContent": "Do you want to edit only this event or entire series?",
"deleteRecurrenceContent": "Do you want to delete only this event or entire series?",
"deleteContent": "Are you sure you want to delete this event?",
"deleteMultipleContent": "Are you sure you want to delete the selected events?",
"newEvent": "New Event",
"title": "Title",
"location": "Location",
"description": "Description",
"timezone": "Timezone",
"startTimezone": "Start Timezone",
"endTimezone": "End Timezone",
"repeat": "Repeat",
"saveButton": "Save",
"cancelButton": "Cancel",
"deleteButton": "Delete",
"recurrence": "Recurrence",
"wrongPattern": "The recurrence pattern is not valid.",
"seriesChangeAlert": "The changes made to specific instances of this series will be cancelled and those events will match the series again.",
"createError": "The duration of the event must be shorter than how frequently it occurs. Shorten the duration, or change the recurrence pattern in the recurrence event editor."
```

```
"recurrenceDateValidation": "Some months have fewer than the selected date. For these months, the  
occurrence will fall on the last date of the month.",  
"sameDayAlert": "Two occurrences of the same event cannot occur on the same day.",  
"editRecurrence": "Edit Recurrence",  
"repeats": "Repeats",  
"alert": "Alert",  
"startEndError": "The selected end date occurs before the start date.",  
"invalidDateError": "The entered date value is invalid.",  
"ok": "Ok",  
"occurrence": "Occurrence",  
"series": "Series",  
"previous": "Previous",  
"next": "Next",  
"timelineDay": "Timeline Day",  
"timelineWeek": "Timeline Week",  
"timelineWorkWeek": "Timeline Work Week",  
"timelineMonth": "Timeline Month",  
"expandAllDaySection": "Expand",  
"collapseAllDaySection": "Collapse"  
},  
"recurrenceeditor": {  
  "none": "None",  
  "daily": "Daily",  
  "weekly": "Weekly",  
  "monthly": "Monthly",  
  "month": "Month",  
  "yearly": "Yearly",  
  "never": "Never",  
  "until": "Until",  
  "count": "Count",  
  "first": "First",  
  "second": "Second",  
  "third": "Third",
```

```

"fourth": "Fourth",
"last": "Last",
"repeat": "Repeat",
"repeatEvery": "Repeat Every",
"on": "Repeat On",
"end": "End",
"onDay": "Day",
"days": "Day(s)",
"weeks": "Week(s)",
"months": "Month(s)",
"years": "Year(s)",
"every": "every",
"summaryTimes": "time(s)",
"summaryOn": "on",
"summaryUntil": "until",
"summaryRepeat": "Repeats",
"summaryDay": "day(s)",
"summaryWeek": "week(s)",
"summaryMonth": "month(s)",
"summaryYear": "year(s)"
}
}
});
`

```

Setting date format

Scheduler can be used with all valid date formats and by default it follows the universal date format "MM/dd/yyyy". If the `dateFormat` property is not specified particularly, then it will work based on the locale that is assigned to the Scheduler. As the default locale applied on Scheduler is "en-US", this makes it to follow the "MM/dd/yyyy" pattern.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from '../datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    dateFormat: "yyyy/MM/dd",

```

```

views: ['Day', 'Week', 'WorkWeek', 'Month'],
selectedDate: new Date(2018, 1, 15),
eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```


Setting the time format

Time formats is a way of representing the time value in different string formats in the Scheduler. By default, the time mode of the Scheduler can be either 12 or 24 hours format which is completely based on the **locale** set to the Scheduler. Since the default **locale** value of the Scheduler is en-US, the time mode will be set to 12 hours format automatically. You can also customize the format by using the **timeFormat** property. To know more about the time format standards, refer to the [Date and Time Format](#) section.

The following example demonstrates the Scheduler component in 24 hours format.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    timeFormat: 'HH:mm',
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    selectedDate: new Date(2018, 1, 15),
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: **timeFormat** property only accepts the valid time format's.

Displaying Scheduler in RTL mode

The Scheduler layout and its behavior can be changed as per the common RTL (Right to Left) conventions by setting **enableRtl** to **true**. By doing so, the Scheduler will display its usual layout from right to left. It's default value is **false**.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    enableRtl: true,
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

See Also

- [How to change first day of the week in the Scheduler](#)

Accessibility in EJ2 JavaScript Schedule control

The Scheduler has been designed based on the WAI-ARIA specifications, thus applying the appropriate ARIA roles, states and properties for the Scheduler elements. It is also available with a built-in keyboard navigation support, making it easier for the people who use assistive technologies or who completely rely on the Keyboard support. As per the accessibility standard, the navigated dates, views and other interactive actions performed on the Scheduler will be read out to the target users who use assistive technologies such as screen readers.

The Scheduler makes use of the most required ARIA attributes such as `aria-label` and `role` to support the accessibility in it. To be more accurate, it must be used with an ARIA compliant browser along with the screen reader running from backend.

The accessibility compliance for the Schedule control is outlined below.

| Accessibility Criteria | Compatibility |

```

| -- | -- |
| WCAG 2.2 Support |  |
| Section 508 Support |  |
| Screen Reader Support |  |
| Right-To-Left Support |  |
| Color Contrast |  |
| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the control meet the requirement.</div>
<div> - Some features of the control do not meet the requirement.</div>
<div> - The
control does not meet the requirement.</div>

```

ARIA attributes

The Scheduler parent element is assigned with a role of `main`, to denote it as the main content of a control as well as a unique element of the entire document.

The following ARIA attributes are used in the Scheduler.

Attributes	Description
------------	-------------

-----	-----
-------	-------

| role="main" | Attribute added to the Scheduler element describes the actual role of the element and denote it as a main and unique content. |

| role="button" | Attribute is assigned to the appointments of Scheduler, to denote it as a clickable element. |

| aria-label | Attribute is set to the Scheduler parent element and its default value is Scheduler's current date. On every time, the date is navigated, this attribute is updated with appropriate current date values. It is also assigned to other scheduler UI elements such as previous and next date navigation buttons depicting its purpose, div element displaying date range in the header bar and appointment elements. |

| aria-labelledby | It indicates editor dialog title to the user through assistive technologies. |

| aria-describedby | It indicates editor dialog content description to the user through assistive technologies. |

| aria-disabled | Attribute is set to the appointment element to indicates the disabled state of the Scheduler.

Keyboard interaction

All the Scheduler actions can be controlled via keyboard keys by using the `allowKeyboardInteraction` property which is set to `true` by default. The following are the standard keys that work on Scheduler.

Keys	Description
----- -----	
Alt + j	Focuses the Scheduler element [provided from application end].
Tab	Focuses the first or active item on the Scheduler header bar and then move the focus to the next available event elements. If no events present, then focus moves out of the control.
Shift + Tab	Reverse focusing of the <code>Tab</code> key functionality. Inverse focusing of event elements from the last one and then move onto the first or active item on Scheduler header bar and then moves out of the control.
Enter	Opens the quick info popup on the selected cells or events.
Escape	Closes any of the popup that are in open state.
Arrow	To move onto the next available cells in either of the needed directions. (left, right, top and right)
Shift + Arrow	For multiple cell selection on either direction.
Delete	Deletes one or more selected events.
Ctrl + Click on events	To select multiple events.
Alt + Number (from 1 to 6)	To switch between the views of Scheduler.
Ctrl + Left Arrow	To navigate to the previous date period.
Ctrl + Right Arrow	To navigate to the next date period.

| Left or Right Arrow | On pressing any of these keys, when focus is currently on the Schedule header bar, moves the focus to the previous or next items in the header bar. |

| Space or Enter | It activates any of the focused items. |

| Page Up & Page Down | To scroll through the work cells area. |

| Home | To move the selection to the first cell of Scheduler. |

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Ensuring accessibility

The Scheduler control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Scheduler control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Scheduler control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Scheduler styling in EJ2 JavaScript Schedule control

To modify the Scheduler appearance, you need to override the default CSS of Scheduler. Also, you have an option to create your own custom theme using our [Theme Studio](#). Please find the list of CSS classes in Scheduler.

| Css class | Purpose |

|-----|-----|

| .e-schedule .e-vertical-view .e-work-cells | Work cells in vertical views of scheduler |

| .e-schedule .e-month-view .e-work-cells | Work cells in month view of scheduler |

| .e-schedule .e-month-view .e-other-month | Work cells of other month in month view of scheduler |

| .e-schedule .e-timeline-view .e-work-cells | Work cells in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-work-cells | Work cells in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells | Work cells in timeline year view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells.e-other-month | Work cells of other month in timeline year view of scheduler |

| .e-schedule .e-month-agenda-view .e-work-cells | Work cells in month agenda view of scheduler |

| .e-schedule .e-month-agenda-view .e-other-month | Work cells of other month in month agenda view of scheduler |

| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-other-month | Work cells of other month in year view of scheduler |

| .e-schedule .e-vertical-view .e-all-day-cells | All day cells in vertical views of scheduler |

| .e-schedule .e-vertical-view .e-work-hours | Work hour cells in vertical views of scheduler |

| .e-schedule .e-month-view .e-work-days | Work day cells in month view of scheduler |

| .e-schedule .e-month-agenda-view .e-work-days | Work day cells in month agenda view of scheduler |

| .e-schedule .e-timeline-view .e-work-hours | Work hour cells in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-work-days | Work day cells in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-work-cells.e-work-days | Work day cells in timeline year view of scheduler |

| .e-schedule .e-vertical-view .e-day-wrapper .e-appointment | Appointment in vertical views of scheduler |

| .e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-appointment | All day Appointment in vertical views of scheduler |

| .e-schedule .e-month-view .e-appointment | Appointment in month view of scheduler |

| .e-schedule .e-timeline-view .e-appointment | Appointment in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-appointment | Appointment in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-event-table .e-appointment | Appointment in timeline year view of scheduler |

| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-appointment | Appointment in year view of scheduler |

| .e-schedule .e-agenda-view .e-appointment | Appointment in agenda view of scheduler |

| .e-schedule .e-month-agenda-view .e-appointment-indicator | Appointment in month agenda view of scheduler |

| .e-schedule .e-block-appointment | Block appointment in scheduler |

| .e-schedule .e-read-only | Read only appointment in scheduler. |

| e-appointment-border | Appointment which are currently selected, use the appointment class hierarchical based on your views. |

| e-selected-cells | work cells which are currently selected, use the work cell class hierarchical based on your views. |

| e-header-cells | Header cells of scheduler, use the work cells hierarchical based on your views. |

| .e-schedule .e-vertical-view .e-resource-cells| Resource cells in vertical views of scheduler. |

| .e-schedule .e-month-view .e-resource-cells| Resource cells in month view of scheduler. |

| .e-schedule .e-timeline-view .e-resource-cells | Resource cells in timeline views of scheduler. |

| .e-schedule .e-timeline-month-view .e-resource-cells| Resource cells in timeline month view of scheduler. |

| e-parent-node | Parent resource cells in timeline views of scheduler. |

| e-child-node | Child resource cells in timeline views of scheduler. |

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

Frequently asked questions in EJ2 JavaScript Schedule control

In this article, you can find some frequently asked questions and corresponding solutions while getting hands-on experience with scheduler control.

Grouping with empty resources

Grouping without providing any resource data will throw the following problems.

- Normal(vertical) views are rendered, but you are not able to perform CRUD operations
- Timeline views not at all render and shows empty scheduler table

So, we suggest to avoid grouping with empty resources in the scheduler.

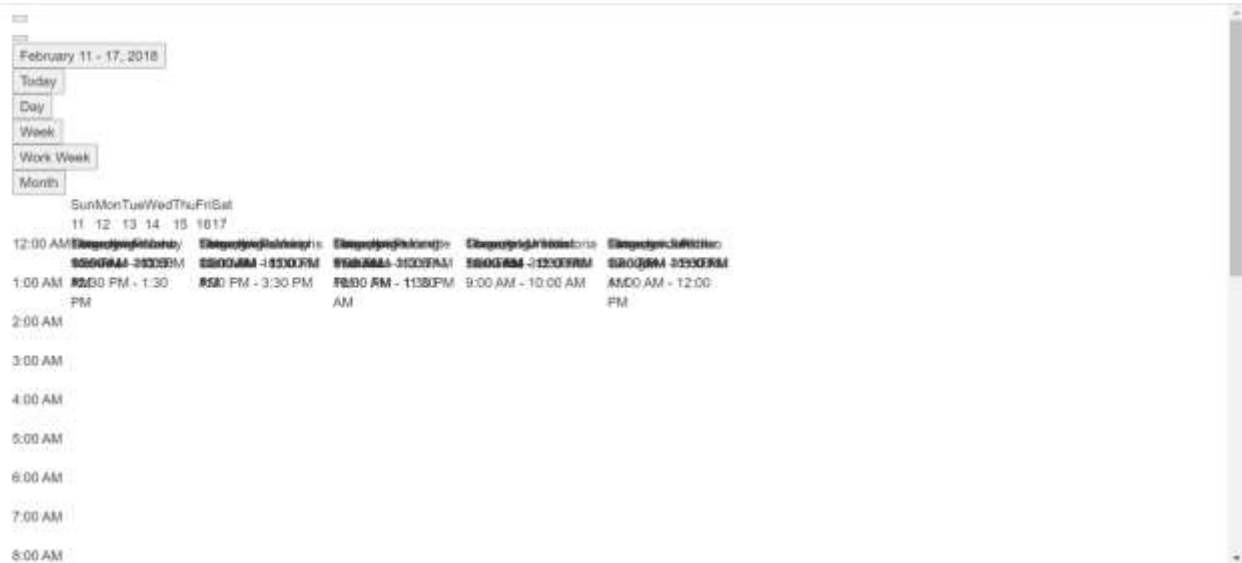
Not providing e-field in editor template

Error: While using editor template, value of `e-field` is missing in editor window.

Solution: `e-field` value is mandatory, we need to add it. Please refer [here](#) for more info.

Missing CSS reference

Error Image:



Solution:

The above problem occurs when missing CSS references for the scheduler in a project. You can resolve this issue by providing proper CSS for the scheduler.

,

```
<html>
```

```
<head>
```

```
<title>Syncfusion Javascript Sample</title>
```



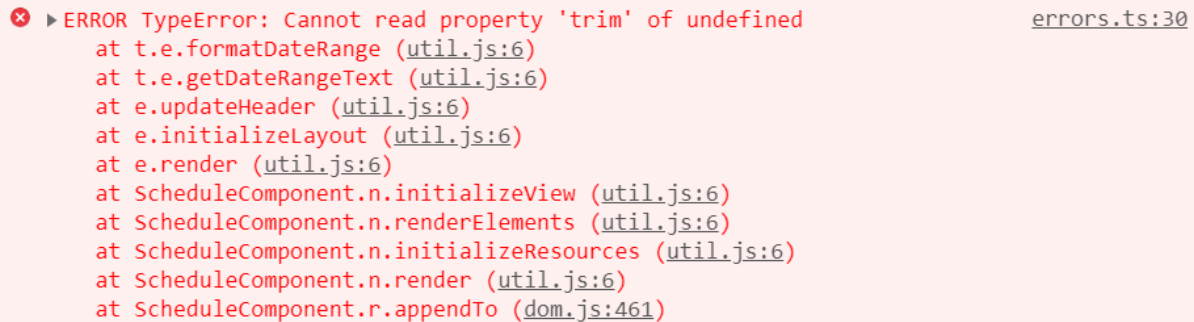
```
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
<meta http-equiv="x-ua-compatible" content="ie=edge">
<meta name="author" content="Syncfusion" />
<! — scheduler CSS is referred from this link —>
<link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet">
</head>
<body class="material">
<div id='sample'>
</body>
</html>
`
```

[QuickInfoTemplate at bottom](#)

When using the `quickInfoTemplate` in scheduler, sometimes quickinfo popup not shown fully at the bottom area of scheduler. You can resolve this by using `cellClick` and `eventClick` events and below code snippet.

```
`javascript
var eventAdded = false;
var scheduleObj = new ej.schedule.Schedule({
.
.
cellClick: onClick,
eventClick: onClick
});
scheduleObj.appendTo('#Schedule');
function onClick(args) {
if (!this.eventAdded) {
let popupInstance = document.querySelector('.e-quick-popup-wrapper').ej2_instances[0];
popupInstance.open = () => {
popupInstance.refreshPosition();
};
this.eventAdded = true;
}
}
```

Not processing culture files while using localization

Error Image:


```

✖ ERROR TypeError: Cannot read property 'trim' of undefined
    at t.e.formatDateRange (util.js:6)
    at t.e.getDateRangeText (util.js:6)
    at e.updateHeader (util.js:6)
    at e.initializeLayout (util.js:6)
    at e.render (util.js:6)
    at ScheduleComponent.n.initializeView (util.js:6)
    at ScheduleComponent.n.renderElements (util.js:6)
    at ScheduleComponent.n.initializeResources (util.js:6)
    at ScheduleComponent.n.render (util.js:6)
    at ScheduleComponent.r.appendTo (dom.js:461)
errors.ts:30

```

While using [locale](#) in scheduler, not processing the `loadCultureFiles` properly throws the problem.

Solution: Properly add the culture files(numberingSystems, timeZoneNames, loadCldr, L10n etc.,) and `loadCultureFiles` method in your project will resolve the problem.

```

`javascript
loadCultureFiles();

var localeTexts;

var localeAjax = new ej.base.Ajax('./locale.json', 'GET', false);

localeAjax.onSuccess = function (value) {
    localeTexts = value;
};

localeAjax.send();

ej.base.L10n.load(JSON.parse(localeTexts));

function loadCultureFiles() {
    // Processing culture files

    var files = ['ca-gregorian.json', 'numbers.json', 'numberingSystems.json', 'timeZoneNames.json'];
    var loader = ej.base.loadCldr;
    var loadCulture = function (prop) {
        var val, ajax;

        ajax = new ej.base.Ajax('./' + files[prop], 'GET', false);
        ajax.onSuccess = function (value) {
            val = value;
        };
        ajax.send();
        loader(JSON.parse(val));
    };
}

```

```

};
for (var prop = 0; prop < files.length; prop++) {
  loadCulture(prop);
}
}
var scheduleObj = new ej.schedule.Schedule({
  width: '100%',
  height: '550px',
  locale: 'fr-CH',
});
scheduleObj.appendTo('#Schedule');
`

```

Getting instance of the Scheduler component

User can access the component instance from the component element through the variable where you are initializing the Schedule component(scheduleObj) or by using the ej2_instances property as shown below.

```

`javascript
var scheduleObj = new ej.schedule.Schedule({
  .
  .
  cellClick: onClick,
});
scheduleObj.appendTo('#Schedule');
function onClick(args) {
  let scheduleInstance = document.querySelector('.e-schedule').ej2_instances[0];
}
`

```

Ej1 api migration in EJ2 JavaScript Schedule control

This topic shows the API equivalent of JS2 Scheduler component to be used, while migrating your project that uses JS1 Scheduler.

Scheduler

Properties

Behavior	API in Essential JS 1	API in Essential JS 2

| To change the display of days count in agenda view | **Property:** *daysInAgenda*

 \$("#Schedule").ejSchedule({
 currentView: "Agenda",
daysInAgenda: 7
}); | **Property:** *agendaDaysCount*

 var scheduleObj = new ej.schedule.Schedule({
currentView: 'Agenda',
agendaDaysCount: 7
});
scheduleObj.appendTo('#Schedule'); |

| Preventing deletion of appointment | **Property:** *allowDelete*

 \$("#Schedule").ejSchedule({
allowDelete: false
}); | Not applicable |

| Allows dragging and dropping of appointments | **Property:** *allowDragAndDrop*

\$("#Schedule").ejSchedule({
 allowDragAndDrop: false
}); | **Property:** *allowDragAndDrop*

 var scheduleObj = new ej.schedule.Schedule({
allowDragAndDrop: false
});
scheduleObj.appendTo('#Schedule'); |

| Enabling inline editing of appointments | **Property:** *allowInline*

\$("#Schedule").ejSchedule({
 allowInline: true
}); | Not applicable |

| Allow keyboard interactions | **Property:** *allowKeyboardNavigation*

 \$("#Schedule").ejSchedule({
allowKeyboardNavigation: false
}); | **Property:** *allowKeyboardInteraction*

 var scheduleObj = new ej.schedule.Schedule({
allowKeyboardInteraction: false
});
scheduleObj.appendTo('#Schedule'); |

| Enable resizing of appointments | **Property:** *enableAppointmentResize*

 \$("#Schedule").ejSchedule({
enableAppointmentResize: true
}); | **Property:** *allowResizing*

 var scheduleObj = new ej.schedule.Schedule({
allowResizing: true
});
scheduleObj.appendTo('#Schedule'); |

| Blocking time intervals | **Property:** *blockoutSettings*

 \$("#Schedule").ejSchedule({
currentDate: new Date(2014, 4, 5),
blockoutSettings: {
enable: true,
dataSource: [{
BlockId: 101,
BlockStartTime: new Date(2014, 4, 5, 10, 00),
BlockEndTime: new Date(2014, 4, 5, 11, 00),
BlockSubject: "Service",
IsBlockAppointment: true
}],
id: "BlockId",
startTime: "BlockStartTime",
endTime: "BlockEndTime",
subject: "BlockSubject",
isBlockAppointment: "IsBlockAppointment"
}); | **Property:** *isBlock*

 var data = [{ Id: 1, Subject: 'Explosion of Betelgeuse Star', StartTime: new Date(2018, 1, 15, 9, 30), EndTime: new Date(2018, 1, 15, 11, 0), IsBlock: true }
var scheduleObj = new ej.schedule.Schedule({
eventSettings: { *dataSource*: data }
});
scheduleObj.appendTo('#Schedule'); |

| Categorizing the appointments | **Property:** *categorizeSettings*

 \$("#Schedule").ejSchedule({
categorizeSettings: {
enable: false,
allowMultiple: false,
appointmentSettings: {
dataSource: [{
id: 100,
subject: "Testing",
StartTime: new Date(2014, 4, 2, 9, 00),
EndTime: new Date(2014, 4, 2, 10, 30),
EventCategorize: "6,4,3"}],
 categorize: "EventCategorize"
}); | Not applicable |

| Setting cell height | **Property:** *cellHeight*

 \$("#Schedule").ejSchedule({
cellHeight: "30px"
}); | Not applicable |

| Cell template | **Property:** *workCellsTemplateld*

 \$("#Schedule").ejSchedule({
allDayCellsTemplate:
 "#allDayTemplate",
workCellsTemplateld: "#workTemplate"
}); | **Property:** *cellTemplate*

 var scheduleObj = new ej.schedule.Schedule({
cellTemplate:
 "#cellTemplate";
});
scheduleObj.appendTo('#Schedule'); |

| Setting cell width | **Property:** *cellWidth*

 \$("#Schedule").ejSchedule({
cellWidth:
 "40px"
}); | Not applicable |

| CSS class | **Property:** *cssClass*

 \$("#Schedule").ejSchedule({
cssClass:
 "customStyle"
}); | **Property:** *cssClass*

var scheduleObj = new
 ej.schedule.Schedule({
cssClass: "customStyle"
});

 scheduleObj.appendTo('#Schedule'); |

| Enabling Context-menu option | **Property:** *contextMenuSettings*

 \$("#Schedule").ejSchedule({
contextMenuSettings: {
 enable: true,
 menuItems:{

 appointment:[
{ id: "open", text: "Open Appointment" },
{ id: "delete",
text: "Delete
 Appointment" }
],
cells: [
{ id: "new", text: "New Appointment"},

id:
 "recurrence", text: "New Recurring Appointment" }
]
}
}); | Not applicable |

| Current view | **Property:** *currentView*

 \$("#Schedule").ejSchedule({
currentView:
 ej.Schedule.CurrentView.Week
}); | **Property:** *currentView*

 var scheduleObj = new
 ej.schedule.Schedule({
currentView: 'Week'
});

 scheduleObj.appendTo('#Schedule'); |

| Date format | **Property:** *dateFormat*

 \$("#Schedule").ejSchedule({
 dateFormat:
 "yyyy/MM/dd"
}); | **Property:** *dateFormat*

 var scheduleObj = new
 ej.schedule.Schedule({
dateFormat: "yyyy/MM/dd"
});

 scheduleObj.appendTo('#Schedule'); |

| Date header template | **Property:** *dateHeaderTemplateld*

 \$("#Schedule").ejSchedule({
dateHeaderTemplateld: "#dateHeaderTemplate"
}); |
Property: *dateHeaderTemplate*

 var scheduleObj = new
 ej.schedule.Schedule({
dateHeaderTemplate: "#dateHeaderTemplate"
});

 scheduleObj.appendTo('#Schedule'); |

| Editor template | Not Applicable | **Property:** *editorTemplate*

 var scheduleObj = new
 ej.schedule.Schedule({
editorTemplate: "#editorTemplate"
});

 scheduleObj.appendTo('#Schedule'); |

| Enable load on demand | **Property:** *enableLoadOnDemand*

 \$("#Schedule").ejSchedule({
enableLoadOnDemand: true
}); | Not applicable |

| Enable persistence | **Property:** *enablePersistence*

 \$("#Schedule").ejSchedule({
enablePersistence: true
}); | **Property:** *enablePersistence*

 var scheduleObj = new ej.schedule.Schedule({
 enablePersistence: true
}); |

| Enable RTL | **Property:** *enableRTL*

 \$("#Schedule").ejSchedule({
enableRTL:
 true
}); | **Property:** *enableRTL*

 var scheduleObj = new ej.schedule.Schedule({

 enableRTL: true
});
 scheduleObj.appendTo('#Schedule'); |

| Setting end hour of the scheduler | **Property:** *endHour*

 `$("#Schedule").ejSchedule({
endHour: 18
});` | **Property:** *endHour*

 `var scheduleObj = new ej.schedule.Schedule({
endHour: '20:00'
});
 scheduleObj.appendTo('#Schedule');` |

| Setting first day of the week | **Property:** *firstDayOfWeek*

 `$("#Schedule").ejSchedule({
firstDayOfWeek: "Monday"
});` | **Property:** *firstDayOfWeek*

 `var scheduleObj = new ej.schedule.Schedule({
 firstDayOfWeek: 1
});
 scheduleObj.appendTo('#Schedule');` |

| Height of the scheduler | **Property:** *height*

 `$("#Schedule").ejSchedule({
height: "550px"
});` | **Property:** *height*

 `var scheduleObj = new ej.schedule.Schedule({
height: '550px'
});
 scheduleObj.appendTo('#Schedule');` |

| Locale | **Property:** *locale*

 `$("#Schedule").ejSchedule({
 locale: "fr-FR"
});` | **Property:** *locale*

 `var scheduleObj = new ej.schedule.Schedule({
 locale: 'fr-FR'
});
 scheduleObj.appendTo('#Schedule');` |

| Priority settings for appointments | **Property:** *PrioritySettings*

 `$("#Schedule").ejSchedule({
PrioritySettings: {
enable: true,
dataSource: [
text: "text",
value: "value"
}}};` | Not applicable |

| Read only | **Property:** *readOnly*

 `$("#Schedule").ejSchedule({
 readOnly: true
});` | **Property:** *readOnly*

 `var scheduleObj = new ej.schedule.Schedule({
readOnly: true
});
 scheduleObj.appendTo('#Schedule');` |

| Reminder settings | **Property:** *reminderSettings*

 `$("#Schedule").ejSchedule({
 reminderSettings: {
enable: true,
alertBefore: 10
}}};` | Not applicable |

| Resource header template | **Property:** *resourceHeaderTemplateId*

 `$("#Schedule").ejSchedule({
resourceHeaderTemplateId: "#resTemplate",
group: {
resources: ["Rooms"]
},
resources: [{
field: "roomId",title: "Room",name: "Rooms", allowMultiple: false,
resourceSettings: {
 dataSource: [{
{ text: "ROOM", id: 2, color: "#56ca85"}
,
text: "text", id: "id", color: "color"}
}],
}}];` | **Property:** *resourceHeaderTemplate*

 `var scheduleObj = new ej.schedule.Schedule({
 resourceHeaderTemplate: '#resTemplate',
group: {
resources: ['Owners']
},
resources: [{
field: 'TaskId', title: 'Assignee',name: 'Owners', allowMultiple: true,
dataSource: [{
{ text: 'Smith', id: 2, color: '#7fa900'}
,
textField: 'text', idField: 'id', colorField: 'color'}
}],
}}];
 scheduleObj.appendTo('#Schedule');` |

| Current date of the scheduler | **Property:** *currentDate*

 `$("#Schedule").ejSchedule({
currentDate: new Date(2014,4,5)
});` | **Property:** *selectedDate*

 `var scheduleObj = new ej.schedule.Schedule({
 selectedDate: new Date(2014,4,5)
});
 scheduleObj.appendTo('#Schedule');` |

| Show all day row | **Property:** *showAllDayRow*

 `$("#Schedule").ejSchedule({
 showAllDayRow: false
});` | Not applicable |

| Show appointment navigator | **Property:** *showAppointmentNavigator*

 \$("#Schedule").ejSchedule({
 showAppointmentNavigator: false
}); | Not applicable |

| Show delete confirmation dialog | **Property:** *showDeleteConfirmationDialog*

 \$("#Schedule").ejSchedule({
 showDeleteConfirmationDialog: false
}); | Not applicable |

| Show header bar | **Property:** *showHeaderBar*

 \$("#Schedule").ejSchedule({

 showHeaderBar: false
}); | **Property:** *showHeaderBar*

 var scheduleObj = new
 ej.schedule.Schedule({
showHeaderBar: false
});

 scheduleObj.appendTo('#Schedule'); |

| Show location field in event window | **Property:** *showLocationField*

 \$("#Schedule").ejSchedule({
 showLocationField: false
}); | Not applicable |

| Show time zone fields in event window | **Property:** *showTimeZoneFields*

 \$("#Schedule").ejSchedule({
 showTimeZoneFields: false
}); | Not applicable |

| Show previous and next month dates in month view | **Property:** *showNextPrevMonth*

 \$("#Schedule").ejSchedule({
showNextPrevMonth: false
}); | Not applicable |

| Show overflow button | **Property:** *showOverflowButton*

 \$("#Schedule").ejSchedule({
showOverflowButton: false
}); | **Property:** *rowAutoHeight*

 var scheduleObj = new ej.schedule.Schedule({
 rowAutoHeight: true
});

 scheduleObj.appendTo('#Schedule'); |

| Show quick popup | **Property:** *showQuickWindow*

 \$("#Schedule").ejSchedule({
showQuickWindow: false
}); | **Property:** *showQuickInfo*

 var scheduleObj = new ej.schedule.Schedule({
showQuickInfo: false
});

 scheduleObj.appendTo('#Schedule'); |

| Show current time indicator | **Property:** *showCurrentTimeIndicator*

 \$("#Schedule").ejSchedule({
showCurrentTimeIndicator: false
}); | **Property:**
showTimeIndicator

 var scheduleObj = new
 ej.schedule.Schedule({
showTimeIndicator: false
});

 scheduleObj.appendTo('#Schedule'); |

| Show week number | Not Applicable | **Property:** *showWeekNumber*

 var scheduleObj =
 new ej.schedule.Schedule({
 showWeekNumber: false
});

 scheduleObj.appendTo('#Schedule'); |

| Show weekend days | **Property:** *showWeekend*

 \$("#Schedule").ejSchedule({
showWeekend: false
}); | **Property:** *showWeekend*

 var scheduleObj = new ej.schedule.Schedule({
 showWeekend: false
});

 scheduleObj.appendTo('#Schedule'); |

| Setting start hour of the scheduler | **Property:** *startHour*

 \$("#Schedule").ejSchedule({
startHour:9
}); | **Property:** *startHour*

 var
 scheduleObj = new ej.schedule.Schedule({
 startHour:'09:00'
});

 scheduleObj.appendTo('#Schedule'); |

| Setting time mode on scheduler | **Property:** *timeMode*

 \$("#Schedule").ejSchedule({
timeMode:ej.Schedule.TimeMode.Hour24
}); | not applicable |

| Setting timezone for scheduler | **Property:** *timeZone*

 \$("#Schedule").ejSchedule({
timeZone: "UTC -5:00"
}); | **Property:** *timezone*

 var scheduleObj = new ej.schedule.Schedule({
 timezone:'UTC'
});

 scheduleObj.appendTo('#Schedule'); |

| Views in scheduler | **Property:** *views*

 \$("#Schedule").ejSchedule({
views: ["Day", "Week", "WorkWeek", "Month"]
}); | **Property:** *views*

 var scheduleObj = new ej.schedule.Schedule({
views: ['Day','Week','WorkWeek','Month']
});

 scheduleObj.appendTo('#Schedule'); |

| Width of the scheduler | **Property:** *width*

 \$("#Schedule").ejSchedule({
width: "100%"
}); | **Property:** *width*

 var scheduleObj = new ej.schedule.Schedule({
 width:'100%'
});

 scheduleObj.appendTo('#Schedule'); |

| Working days | **Property:** *workWeek*

 \$("#Schedule").ejSchedule({
workWeek:["Tuesday", "Wednesday", "Thursday", "Friday","Saturday"]
}); | **Property:** *workDays*

 var scheduleObj = new ej.schedule.Schedule({
 workDays:[1,3,5]
});

 scheduleObj.appendTo('#Schedule'); |

| Working hours | **Property:** *workHours*

 \$("#Schedule").ejSchedule({
workHours:
highlight: true,
start: 8,
end: 18

}); | **Property:** *workHours*

 var scheduleObj = new ej.schedule.Schedule({
workHours: { highlight: true, start: '08:00', end: '18:00' }
});

 scheduleObj.appendTo('#Schedule'); |

Resources

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| To group the resources in scheduler layout | **Property:** *resources*

 \$("#Schedule").ejSchedule({
group: {
resources: ["Rooms"]
},
resources: {
field: "roomId",title: "Room",name: "Rooms", allowMultiple: false,
resourceSettings: {
 dataSource: [
{ text: "ROOM", id: 2, color: "#56ca85"}],
text: "text", id: "id", color: "color"}
},
}); | **Property:** *resources*

 var scheduleObj = new ej.schedule.Schedule({
group: {
resources: ['Owners']
},
resources: [{
field: 'TaskId', title: 'Assignee',name: 'Owners', allowMultiple: true,
dataSource: [
{ text: 'Smith', id: 2, color: '#7fa900' }
},
textField: 'text', idField: 'id', colorField: 'color'}],
});

 scheduleObj.appendTo('#Schedule'); |

| Allowing multiple selection of resources in event window | **Property:** *allowMultiple*

 \$("#Schedule").ejSchedule({
group: {
resources: ["Rooms"]
},
resources: {
field: "roomId",title: "Room",name: "Rooms", allowMultiple: false,
resourceSettings: {
 dataSource: [
{ text: "Room3", id: 3, groupId: 2, color: "#56ac88", start: "10", end: "15", workWeek: ["sunday","saturday"] }],
text: "text", id: "id",


```
color: "color", groupId: "groupId", start: "start", end: "end", workWeek:
"workWeek"]<br>]],<br>}); | Property: allowMultiple <br><br> var scheduleObj = new
ej.schedule.Schedule({<br>group: {<br>resources: ['Projects', 'Categories']<br>,<br>resources:
[<br><br>field: 'ProjectId', title: 'Choose Project', name: 'Projects',<br>dataSource: [<br>{ text:
'PROJECT 1', id: 2, color: '#56ca85' }<br>],<br>textField: 'text', idField: 'id', colorField: 'color'},
{<br>field: 'CategoryId', title: 'Category',name: 'Categories', allowMultiple: true,<br>dataSource:
[<br>{ text: 'Testing', id: 2, color: '#7fa900' }],<br>textField: 'text', idField: 'id', colorField:
'color'}]],<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

```
| Setting different work hours for each resource | <br><br> $("#Schedule").ejSchedule({<br>group:
{<br>resources: ["Rooms,Owners"]<br>,<br>resources: [{<br>field: "roomId",title:
"Room",name: "Rooms", allowMultiple: false,<br>resourceSettings: {<br>dataSource: [<br>{
text: "Room3", id: 3, groupId: 2, color: "#56ac88", start: "10", end: "15", workWeek:
["sunday","saturday"] }],<br>text: "text", id: "id", groupId: "groupId", color: "color", start:
"WorkHourStart" end: "WorkHourEnd" workWeek: "CustomDays"<br>,<br>resources:
[{<br>field: "ownerId",title: "Owner",name: "Owners", allowMultiple:
false,<br>resourceSettings: {<br>dataSource: [<br>{ text: "Margaret", id: 1, groupId: 1, color:
'#1aaa55' }],<br>text: "text", id: "id", groupId: "groupId", color: "color"<br>}<br>]],<br>}); | var
scheduleObj = new ej.schedule.Schedule({<br>group: {<br>resources: ['Projects',
'Categories']<br>,<br>resources: [<br><br>field: 'ProjectId', title: 'Choose Project', name:
'Projects',<br>dataSource: [<br>{ text: "Project3", id: 3, groupId: 2, color: "#56ac88", start:
"10:00", end: "15:00", workWeek: [1, 2] }],<br>textField: 'text', idField: 'id', colorField: 'color'},
{<br>field: 'CategoryId', title: 'Category',name: 'Categories', allowMultiple: true,<br>dataSource:
[<br>{ text: "Testing", id: 3, groupId: 2, color: "#56ac88", start: "10:00", end: "15:00",
workWeek: [2,6] }],<br>textField: 'text', idField: 'id', colorField: 'color', workDaysField:
'workDays' startHourField: 'startHour', endHourField: 'endHour'}]],<br>views: ['Week',
'Month']<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

Group

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```
| Resources | Property: resources <br><br> $("#Schedule").ejSchedule({<br>group:
{<br>resources: ["Rooms"]<br>,<br>resources: [<br><br>field: "roomId",title: "Room",name:
"Rooms", allowMultiple: false,<br>resourceSettings: {<br>dataSource: [<br>{ text: "ROOM", id:
2, color: "#56ca85"}],<br>text: "text", id: "id", color: "color"<br>}<br>]],<br>}); | Property: resources
<br><br> var scheduleObj = new ej.schedule.Schedule({<br>group: {<br>resources:
['Owners']<br>,<br>resources: [{<br>field: 'TaskId', title: 'Assignee',name: 'Owners',
allowMultiple: true,<br>dataSource: [<br>{ text: 'Smith', id: 2, color: '#7fa900'
}<br>],<br>textField: 'text', idField: 'id', colorField:
'color'}]],<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

```
| Allow group editing | Property: allowGroupEditing <br><br>
$("#Schedule").ejSchedule({<br>group: {<br>resources: ["Rooms"],<br>allowGroupEditing:
true<br>,<br>resources: [<br><br>field: "roomId",title: "Room",name: "Rooms", allowMultiple:
false,<br>resourceSettings: {<br>dataSource: [<br>{ text: "ROOM1", id: 1,color: "#f8a398"
```

```
},<br>{ text: "ROOM2", id: 2, color: "#56ca85"}],<br>text: "text", id: "id", color:
"color"}<br>]],<br>}); | Property: allowGroupEdit <br><br> var scheduleObj = new
ej.schedule.Schedule({<br>group: {<br>resources: ['Owners'],<br>allowGroupEdit:
true<br>},<br>resources: [{<br>field: 'TaskId', title: 'Assignee',name: 'Owners', allowMultiple:
true,<br>dataSource: [<br>{ text: 'Smith', id: 2, color: '#7fa900' }<br>],<br>textField: 'text',
idField: 'id', colorField: 'color'}],<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

```
| Grouping resources by date | Not applicable | Property: byDate <br><br> var scheduleObj = new
ej.schedule.Schedule({<br>group: {<br>byDate: true,<br>resources:
['Owners']<br>},<br>resources: [{<br>field: 'TaskId', title: 'Assignee',name: 'Owners',
allowMultiple: true,<br>dataSource: [<br>{ text: 'Smith', id: 2, color: '#7fa900'
}<br>],<br>textField: 'text', idField: 'id', colorField:
'color'}],<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

```
| Grouping resources based on its group ID | Not applicable | Property: byGroupId <br><br> var
scheduleObj = new ej.schedule.Schedule({<br>group: {<br>byGroupId: true,<br>resources:
['Rooms', 'Owners']<br>},<br>resources: [{<br>field: 'RoomId', title: 'Room',name: 'Rooms',
allowMultiple: false,<br>dataSource: [<br>{ text: 'Room 1', id: 1, color: '#7fa900' },<br>{ text:
'Room 2', id: 2, color: '#56ca85' }<br>],<br>textField: 'text', idField: 'id', colorField:
'color'}],<br>byGroupId: true,<br>resources: ['Rooms', 'Owners']<br>},<br>resources: [{<br>field:
'OwnerId', title: 'Owner',name: 'Owners', allowMultiple: true,<br>dataSource: [<br>{ text:
'Nancy', id: 1, groupId: 1, color: '#7fa900' },<br>{ text: 'Steven', id: 2, groupId: 2, color: '#56ca85'
},<br>{ text: 'Michael', id: 3, groupId: 2, color: '#7499e1' }<br>],<br>textField: 'text', idField:
'id',groupIdField: 'groupId', colorField: 'color'}],<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

```
| Enabling compact view on mobile mode | Not applicable | Property: enableCompactView <br><br>
var scheduleObj = new ej.schedule.Schedule({<br>group: {<br>enableCompactView:
false,<br>resources: ['Owners']<br>},<br>resources: [{<br>field: 'TaskId', title: 'Assignee',name:
'Owners', allowMultiple: true,<br>dataSource: [<br>{ text: 'Smith', id: 2, color: '#7fa900'
}<br>],<br>textField: 'text', idField: 'id', colorField:
'color'}],<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

```
| Header tooltip template | Not applicable | Property: headerTooltipTemplate <br><br> var
scheduleObj = new ej.schedule.Schedule({<br>group: {<br>headerTooltipTemplate:
'#resourceHeader',<br>resources: ['Owners']<br>},<br>resources: [{<br>field: 'TaskId', title:
'Assignee',name: 'Owners', allowMultiple: true,<br>dataSource: [<br>{ text: 'Smith', id: 2, color:
'#7fa900' }<br>],<br>textField: 'text', idField: 'id', colorField:
'color'}],<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

Header Rows

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```
| Adding custom rows in the header in timeline views | Not applicable | Property: HeaderRows
<br><br> var scheduleObj = new ej.schedule.Schedule({<br>headerRows: [<br>{ option:
'Month', template: '#month-template' },<br>{ option: 'Week', template: '#week-template'
```

```
},<br>{ option: 'Date' }<br>],<br>views: [<br>{ option: 'TimelineMonth', interval: 12
}],<br>});<br>scheduleObj.appendTo('#Schedule');|
```

TimeScale

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Enabling time scale | **Property:** *enable*


```
$("#Schedule").ejSchedule({<br>timeScale:<br>enable: true<br>}<br>}); | Property: enable
<br><br> var scheduleObj = new ej.schedule.Schedule({<br>timeScale:<br>enable:
true<br>}<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Setting major interval on time scale | **Property:** *majorSlot*


```
$("#Schedule").ejSchedule({<br>timeScale:<br>enable: true,<br>majorSlot: 60<br>}<br>}); |
Property: interval <br><br> var scheduleObj = new ej.schedule.Schedule({<br>enable:
true,<br>interval: 60<br>}<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Setting slot count on time scale | **Property:** *minorSlotCount*


```
$("#Schedule").ejSchedule({<br>timeScale:<br>enable: true,<br>majorSlot:
60,<br>minorSlotCount:3<br>}<br>}); | Property: slotCount <br><br> var scheduleObj = new
ej.schedule.Schedule({<br>timeScale:<br>enable: true,<br>interval: 60,<br>slotCount:
6<br>}<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Defining major slot template | **Property:** *majorSlotTemplateId*


```
$("#Schedule").ejSchedule({<br>timeScale:<br>enable: true,<br>majorSlot:
60,<br>minorSlotCount:6,<br>majorSlotTemplateId: "#majorSlotTemplate"<br>}<br>}); |
Property: majorSlotTemplate <br><br> var scheduleObj = new
ej.schedule.Schedule({<br>timeScale:<br>enable: true,<br>interval: 60,<br>slotCount:
6,<br>majorSlotTemplate:
'#majorSlotTemplate'<br>}<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Defining minor slot template | **Property:** *minorSlotTemplateId*


```
$("#Schedule").ejSchedule({<br>timeScale:<br>enable: true,<br>majorSlot:
60,<br>minorSlotCount:6,<br>minorSlotTemplateId: "#minorSlotTemplate"<br>}<br>}); |
Property: minorSlotTemplate <br><br> var scheduleObj = new ej.schedule.Schedule({<br>enable:
true,<br>interval: 60,<br>slotCount: 6,<br>minorSlotTemplate:
'#minorSlotTemplate'<br>}<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

Quick info templates

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Template for quick popup | Not applicable | **Property:** *quickInfoTemplates*

 var
scheduleObj = new ej.schedule.Schedule({
quickInfoTemplates: {
header:
'#headerTemplate',
content:'#contentTemplate',
footer:
'#footerTemplate'
}
});
scheduleObj.appendTo('#Schedule'); |

Event settings

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Datasource for events | **Property:** *dataSource*


```
$("#Schedule").ejSchedule({<br>appointmentSettings: {<br>dataSource: Appoint,<br>id:
"Id",<br>subject: "Subject",<br>startTime: "StartTime",<br>endTime:
"EndTime",<br>description: "Description",<br>allDay: "AllDay",<br>recurrence:
"Recurrence",<br>recurrenceRule: "RecurrenceRule"<br><br>}); | Property: dataSource
<br><br> var scheduleObj = new ej.schedule.Schedule({<br><br> eventSettings: { dataSource:
data }<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Appointment fields | \$("#Schedule").ejSchedule({
appointmentSettings: {
id:

```
"Id",<br>subject: "Subject",<br>startTime: "StartTime",<br>endTime:
"EndTime",<br>description: "Description",<br>allDay: "AllDay",<br>recurrence:
"Recurrence",<br>recurrenceRule: "RecurrenceRule"<br><br>}); | Property: eventsSettings
<br><br> var scheduleObj = new ej.schedule.Schedule({<br> dataSource: data,<br>fields:
{<br>id: 'id',<br>subject: { name: 'Subject', default: 'New title', title: 'New Event', validation:
ValidationRules },<br>location: { name: 'Location' },<br>description: {name: 'Description'
},<br>startTime: {name: 'StartTime' },<br>endTime: {name: 'EndTime' },<br>recurrenceID:
{name: 'Recurrence' },<br>recurrenceRule : {name: 'Recurrence' },<br>recurrenceException:
{name: 'EndTime' },<br>startimezone: {name: 'StartTime' },<br>endimezone: {name:
'EndTime' }<br><br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Enabling tooltip for appointments | **Property:** *enable*


```
$("#Schedule").ejSchedule({<br>tooltipSettings: {<br>enable: true<br><br>});<br> Note: Here
tooltip setting for events is maintained separately | Property: enableTooltip <br><br> var
scheduleObj = new ej.schedule.Schedule({<br>eventSettings: {<br>dataSource:
data,<br>enableTooltip: true<br><br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Tooltip template for appointments | **Property:** *templateId*


```
$("#Schedule").ejSchedule({<br>tooltipSettings: {<br>enable:
true,<br>templateId:"#tooltip"<br><br>});<br>Note: Here tooltip setting for events is
maintained separately | Property: tooltipTemplate <br><br> var scheduleObj = new
ej.schedule.Schedule({<br>eventSettings: {<br>dataSource: data,<br>enableTooltip:
true,<br>tooltipTemplate: '#template'<br><br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Template for appointments | **Property:** *appointmentTemplateId*


```
$("#Schedule").ejSchedule({<br>appointmentTemplateId:"#appTemplate",<br>appointmentSett
ings: {<br>dataSource: data<br><br>});<br> Note: Here appointment template is used as simple
API | Property: template <br><br> var scheduleObj = new
ej.schedule.Schedule({<br>eventSettings: {<br>dataSource: data,<br>template:
'#eventTemplate'<br><br>});<br>scheduleObj.appendTo('#Schedule'); |
```

| Query | **Property:** *query*

 var dataManager = ej.DataManager({
url:

```
"http://mvc.syncfusion.com/OdataServices/Northwnd.svc",<br>crossDomain: true<br>});<br>
var query =
ej.Query().from("Events").take(10);<br>$("#Schedule").ejSchedule({<br>appointmentSettings:
{<br>dataSource: dataManager,<br>query: query<br><br>}); | Property: query <br><br>var
dataManger = new DataManager({<br>url:'https://ej2services.syncfusion.com/production/web-
```

```
services/api/Schedule',<br>adaptor: new ODataAdaptor});<br> var scheduleObj = new
ej.schedule.Schedule({<br> eventSettings: { dataSource: dataManger, query: new
Query().addParams('ej2schedule', 'true')}}<br>});<br>scheduleObj.appendTo('#Schedule'); |
| Define which resource level color applied to events | Not applicable | Property: resourceColorField
<br><br> var scheduleObj = new ej.schedule.Schedule({<br>group: {<br> resources: ['Projects',
'Categories']<br>},<br>resources: [<br>{<br>field: 'ProjectId', title: 'Choose Project', name:
'Projects',<br>dataSource: [<br>{ text: 'PROJECT 1', id: 2, color: '#56ca85' }<br>],<br>textField:
'text', idField: 'id', colorField: 'color'}, {<br>field: 'CategoryId', title: 'Category',name:
'Categories', allowMultiple: true,<br>dataSource: [<br>{ text: 'Testing', id: 2, color: '#7fa900'
}],<br>textField: 'text', idField: 'id', colorField: 'color'}],<br>eventSettings: { resourceColorField:
'Projects' }<br>});<br>scheduleObj.appendTo('#Schedule'); |
```

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| To add appointments manually | **Method:** saveAppointment()

\$('#Schedule').ejSchedule();
 var data = {
 Id: 1,
Subject: "Testing",
StartTime: new Date(2014, 4, 5, 10, 00),
EndTime: new Date(2014, 4, 5, 12, 00)};
 var scheduleobj = \$("#Schedule").data("ejSchedule");
scheduleobj.saveAppointment(data);
 | **Method:** addEvent()

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
scheduleObj.addEvent({
Id: 1,
Subject: "New Event",
StartTime: new Date(2014, 4, 5, 10, 00),
EndTime: new Date(2014, 4, 5, 12, 00)}
});
 |

| To add resources dynamically | **Method:** addResource()

\$("#Schedule").ejSchedule();
 var data = { text: "Paul", id: 1, groupId: 3, color: "#cc99ff" };
 var index = 0;
 var scheduleobj = \$("#Schedule").data("ejSchedule");
 scheduleobj.addResource(data, "Owners", index); | **Method:** addResource()

 var data = { text: "Paul", id: 1, groupId: 3, color: "#cc99ff" };
 var index = 0;
 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleobj.addResource(data, "Owners", index); |

| databind | Not applicable | **Method:** dataBind()

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleobj.currentView = 'Month';
 scheduleobj.dataBind();
 |

| Delete appointment manually | **Method:** deleteAppointment()

\$("#Schedule").ejSchedule();
 var data = {
 Id: 1,
Subject: "Testing",
StartTime: new Date(2014, 4, 5, 10, 00),
EndTime: new Date(2014, 4, 5, 12, 00)};
 var scheduleobj = \$("#Schedule").data("ejSchedule");
schObj.deleteAppointment(data); | **Method:** deleteEvent()

 var data = {
Id: 1,
Subject: "Testing",
StartTime: new Date(2014, 4, 5, 10, 00),
EndTime: new Date(2014, 4, 5, 12, 00)};
var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleobj.deleteEvent(data); |

| destroy scheduler | **Method:** *destroy()*

\$(("#Schedule").ejSchedule());
 var
scheduleobj = \$(("#Schedule").data("ejSchedule"));
schObj.destroy(); | **Method:** *destroy()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleObj.destroy(); |

| Get cell details | **Method:** *getSlotByElement()*

 \$(("#Schedule").ejSchedule());
 var
scheduleobj = \$(("#Schedule").data("ejSchedule"));
 var \$td = \$(".e-draggableworkarea table
tr td").first();
var slotDetails = obj.getSlotByElement(\$td); | **Method:** *getCellDetails()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var td
= document.querySelector(".e-work-cells");
var cellDetail =
scheduleobj.getCellDetails(td);
 |

| Get current view appointments | **Method:** *getCurrentViewAppointments()*

\$(("#Schedule").ejSchedule());
 var scheduleobj =
\$(("#Schedule").data("ejSchedule"));
 var currApp=
scheduleobj.getCurrentViewAppointments(); | **Method:** *getCurrentViewEvents()*

 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var currApp =
scheduleobj.getCurrentViewEvents(); |

| Get entire appointment collection | **Method:** *getAppointments()*

\$(("#Schedule").ejSchedule());
var scheduleobj = \$(("#Schedule").data("ejSchedule"));
 var
AppDetails = scheduleobj.getAppointments();
 | **Method:** *getEvents()*

 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var AppDetails =
scheduleobj.getEvents(); |

| Get current view dates | Not applicable | **Method:** *getCurrentViewDates()*

 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var AppDetails =
scheduleobj.getCurrentViewDates(); |

| Get event details | Not applicable | **Method:** *getEventDetails()*

 var scheduleobj=
document.getElementById('schedule').ej2_instances[0];
 var AppDetails =
scheduleobj.getEventDetails(appElement);
 |

| Get occurrences using event ID | Not applicable | **Method:** *getOccurrencesById()*

 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var recCollection =
scheduleobj.getOccurrencesById(1);
 |

| Get occurrences in the provided date range | Not applicable | **Method:** *getOccurrencesByRange()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var
sDate = new Date(2018, 1, 12);
 var eDate = new Date(2018, 1, 17);
 var resCollection =
scheduleobj.getOccurrencesByRange(sDate, eDate); |

| Get resource details using index | Not applicable | **Method:** *getResourceByIndex()*

 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var resCollection =
scheduleobj.getResourceByIndex(2); |

| Show spinner | Not applicable | **Method:** *showSpinner()*

 var scheduleobj=
document.getElementById('schedule').ej2_instances[0];
 scheduleobj.showSpinner(); |

| Hide spinner | Not applicable | **Method:** *hideSpinner()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleobj.hideSpinner(); |

| Check whether the time slot is available | Not applicable | **Method:** *isSlotAvailable()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var sTime = new Date(2018, 1, 14, 09, 30);
 var eTime = new Date(2018, 1, 14, 10, 30);
 var resCollection = scheduleobj.isSlotAvailable(sTime, eTime); |

| Open the event window manually | Not applicable | **Method:** *openEditor()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
var \$td = document.querySelector(".e-content-table tbody tr td");
var cellDetail = scheduleobj.getCellDetails(\$td);
 scheduleobj.openEditor(cellDetail); |

| Refresh the scheduler | **Method:** *refresh()*

 \$("#Schedule").ejSchedule();
var scheduleobj = \$("#Schedule").data("ejSchedule");
 scheduleobj.refresh(); | **Method:** *refresh()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
scheduleobj.refresh(); |

| Refresh the events | **Method:** *refreshAppointments()*

 \$("#Schedule").ejSchedule();
var scheduleobj = \$("#Schedule").data("ejSchedule");
 scheduleobj.refreshAppointments(); | **Method:** *refreshEvents()*

 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleobj.refreshEvents(); |

| Refresh the scroller | **Method:** *refreshScroller()*

 \$("#Schedule").ejSchedule();
var scheduleobj = \$("#Schedule").data("ejSchedule");
 scheduleobj.refreshScroller(); | Not Applicable |

| To remove resources dynamically | **Method:** *removeResource()*

\$("#Schedule").ejSchedule();
 var resID = 1;
var scheduleobj = \$("#Schedule").data("ejSchedule");
scheduleobj.removeResource(resID, "Owners"); | **Method:** *removeResource()*

 var data = { text: "Paul", id: 3, groupId: 1, color: "#cc99ff" };
var index = 0;
var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleobj.removeResource(index, "Owners",); |

| Export scheduler as PDF | **Method:** *exportSchedule()*

\$("#Schedule").ejSchedule();
 var scheduleobj = \$("#Schedule").data("ejSchedule");
 scheduleobj.exportSchedule("ActionName","null", null);
 | Not Applicable |

| Export scheduler events to ICS | **Method:** *exportSchedule()*

 \$("#Schedule").ejSchedule();
var scheduleobj = \$("#Schedule").data("ejSchedule");
 scheduleobj.exportSchedule("ActionName", "ExportToICS", null);
 | **Method:** *exportToCalendar()*
 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleObj.exportToCalendar(); |

| Import scheduler events from ICS | Not Applicable | **Method:** *importCalendar()*
 var scheduleobj= document.getElementById('schedule').ej2_instances[0];
 scheduleObj.importCalendar(); |

| Export scheduler appointments in Excel file | **Method** *exportToExcel()*

 var scheduleobj= \$(“#schedule”).ejSchedule(instance);
 scheduleobj.exportToExcel(“ActionName”, null, true); | **Method:** *exportToExcel()*
 var scheduleobj= document.getElementById(‘schedule’).ej2_instances[0];
 scheduleObj.exportToExcel(); |

| Print the scheduler | **Method:** *print()*

 \$(“#Schedule”).ejSchedule();
 var scheduleobj = \$(“#Schedule”).data(“ejSchedule”);
 scheduleobj.print();
 | Not Applicable |

| Filter appointments | **Method:** *filterAppointments()*

 \$(“#Schedule”).ejSchedule();
 var scheduleobj = \$(“#Schedule”).data(“ejSchedule”);
 var filter = [{ field: “Subject”, operator: “contains”, value: “with”, predicate: “or” }];
 var filteredApp = scheduleobj.filterAppointments(filter); | Not Applicable |

| Search appointments | **Method:** *searchAppointments()*

 \$(“#Schedule”).ejSchedule();
 var scheduleobj = \$(“#Schedule”).data(“ejSchedule”);
 var searchApp = scheduleobj.searchAppointments(“with”); | Not Applicable |

| To edit appointments manually | Not applicable | **Method:** *saveEvent()*

 var scheduleobj= document.getElementById(‘schedule’).ej2_instances[0];
 scheduleobj.saveEvent({
 Id: 1,
 Subject: ‘Event edited’,
 StartTime: new Date(2018, 7, 31, 10, 30),
 EndTime: new Date(2018, 7, 31, 12, 0)}); |

| Setting work hours | Not applicable | **Method:** *setWorkHours()*

 var scheduleobj= document.getElementById(‘schedule’).ej2_instances[0];
 scheduleobj.setWorkHours([new Date(2017, 9, 5)], ‘04:00’, ‘08:00’); |

| Scrolling to specific time | Not applicable | **Method:** *scrollTo()*

 var scheduleobj= document.getElementById(‘schedule’).ej2_instances[0];
 scheduleobj.scrollTo(‘12:00’); |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Fires on the beginning of each scheduler action | **Event:** *actionBegin*

 \$(“#Schedule”).ejSchedule({
 actionBegin: function OnActionBegin (args){}
 }); | **Event:** *actionBegin*

 var schObj = new ej.schedule.Schedule({
 actionBegin: function onActionBegin(args) {}
 }); |

| Fires on the completion of each scheduler action | **Event:** *actionComplete*

 \$(“#Schedule”).ejSchedule({
 actionComplete: function onActionComplete (args){}
 });
 | **Event:** *actionComplete*

 var schObj = new ej.schedule.Schedule({
 actionComplete: function onActionComplete (args) {}
 }); |

| Fires when the scheduler action gets failed | Not applicable | **Event:** *actionFailure*

 var schObj = new ej.schedule.Schedule({
 actionFailure: function onActionFailure(args) {}
 }); |

| Fires on appointment hover | **Event:** *appointmentHover*

 \$(“#Schedule”).ejSchedule({
 appointmentHover: function onAppointmentHover (args) {}
 }); | Not applicable |

| Fires before an appointment gets created | **Event:** *beforeAppointmentCreate*

 \$("#Schedule").ejSchedule({
 beforeAppointmentCreate: function
 OnBeforeAppointmentCreate (args){}
 }); | **Event:** *actionBegin*

 var schedule = new
 ej.schedule.Schedule({
 actionBegin: function onActionBegin(args) { if(args.requestType ==
 'eventCreate') {} }
 }); |

| Fires before an appointment gets edited | **Event:** *beforeAppointmentChange*

 \$("#Schedule").ejSchedule({
 beforeAppointmentChange: function
 OnBeforeAppointmentChange (args){}
 }); | **Event:** *actionBegin*

 var schedule = new
 ej.schedule.Schedule({
 actionBegin: function onActionBegin(args) { if(args.requestType ==
 'eventChange') {} }
 }); |

| Fires before an appointment gets deleted | **Event:** *beforeAppointmentRemove*

 \$("#Schedule").ejSchedule({
 beforeAppointmentRemove: function
 OnBeforeAppointmentRemove (args){}
 }); | **Event:** *actionBegin*

 var schedule =
 new ej.schedule.Schedule({
 actionBegin: function onActionBegin(args) {
 if(args.requestType == 'eventRemove') {} }
 }); |

| Fires before the context menu opens on scheduler | **Event:** *beforeContextMenuOpen*

 \$("#Schedule").ejSchedule({
 beforeContextMenuOpen: function
 OnBeforeContextMenuOpen (args){}
 }); | Not applicable |

| Fires on cell click | **Event:** *cellClick*

 \$("#Schedule").ejSchedule({
 cellClick: function
 onCellClick (args){}
 }); | **Event:** *cellClick*

 var schObj = new ej.schedule.Schedule({

 cellClick: function onCellClick (args) {}
 }); |

| Fires on cell double click | **Event:** *cellDoubleClick*

 \$("#Schedule").ejSchedule({

 cellDoubleClick: function oncellDoubleClick (args){}
 }); | **Event:** *cellDoubleClick*

 var schObj = new ej.schedule.Schedule({
 cellDoubleClick: function oncellDoubleClick (args)
 {}
 }); |

| Fires on cell hover | **Event:** *CellHover*

 \$("#Schedule").ejSchedule({
 CellHover:
 function onCellHover (args){}
 }); |

| Fires once the scheduler is created | **Event:** *create*

 \$("#Schedule").ejSchedule({

 Create: function onCreate (args){}
 }); | **Event:** *created*

 var schObj = new
 ej.schedule.Schedule({
 created: function oncreated (args) {}
 }); |

| Fires on data binding action | Not applicable | **Event:** *dataBinding*

 var schObj = new
 ej.schedule.Schedule({
 dataBinding: function onDataBinding (args) {}
 }); |

| Fires after the data is bound to the control | Not applicable | **Event:** *dataBound*

 var
 schObj = new ej.schedule.Schedule({
 dataBinding: function onDataBound (args) {}
 }); |

| Fires once the scheduler is destroyed | **Event:** *destroy*

 \$("#Schedule").ejSchedule({

 destroy: function onDestroy (args){}
 }); | **Event:** *destroyed*

 var schObj = new
 ej.schedule.Schedule({
 destroyed: function onDestroyed (args) {}
 }); |

| Fires on event click | **Event:** *appointmentClick*

 \$("#Schedule").ejSchedule({

 appointmentClick: function onAppointmentClick (args) {}
 }); | **Event:** *eventClick*

```
</></> var schObj = new ej.schedule.Schedule({ <br> eventClick: function  
onAppointmentClick(args) {} <br> }); |
```

| Fires on event double click | **Event:** *appointmentDoubleClick*

\$("#Schedule").ejSchedule({
 appointmentDoubleClick: function onappointmentDoubleClick
(args) {}
}); | Not applicable |

| Fires for keyboard actions | **Event:** *keyDown*

 \$("#Schedule").ejSchedule({

keyDown: function onKeyDown (args) {}
}); | Not applicable |

| Fires on context menu item click | **Event:** *menutemClick*

 \$("#Schedule").ejSchedule({

 menutemClick: function onMenuitemClick (args) {}
}); | Not applicable |

| Fires on navigation | **Event:** *navigation*

 \$("#Schedule").ejSchedule({
 navigation:
function onNavigation (args) {}
}); | **Event:** *navigating*

 var schObj = new
ej.schedule.Schedule({
 navigating: function onNavigating(args) {}
 }); |

| Fires on popup open | **Event:** *appointmentWindowOpen*

 \$("#Schedule").ejSchedule({

 appointmentWindowOpen: function onAppointmentWindowOpen (args) {}
}); | **Event:**
popupOpen

 var schObj = new ej.schedule.Schedule({
 popupOpen: function
onPopupOpen(args) {}
 }); |

| Fires on dragging event | **Event:** *drag*

 \$("#Schedule").ejSchedule({
 drag: function
onDrag(args) {}
 }); | **Event:** *drag*

 var schObj = new ej.schedule.Schedule({

drag: function onDrag(args) {}
 }); |

| Fires on drag start | **Event:** *dragStart*

 \$("#Schedule").ejSchedule({
 dragStart:
function onDragStart(args) {}
 }); | **Event:** *dragStart*

 var schObj = new
ej.schedule.Schedule({
 dragStart: function onDragStart(args) {}
 }); |

| Fires on drag stop | **Event:** *dragStop*

 \$("#Schedule").ejSchedule({
 dragStop:
function onDragStop(args) {}
 }); | **Event:** *dragStop*

 var schObj = new
ej.schedule.Schedule({
 dragStop: function onDragStop(args) {}
 }); |

| Fires on overflow button click | **Event:** *overflowButtonClick*

 \$("#Schedule").ejSchedule({

 overflowButtonClick: function onOverflowButtonClick(args) {}
 }); | **Event:** *popupOpen*

 var schedule = new ej.schedule.Schedule({
 popupOpen: function
onPopupOpen(args) { if(args.requestType == 'EventContainer') {} }
 }); |

| Fires on overflow button hover | **Event:** *overflowButtonHover*

\$("#Schedule").ejSchedule({
 overflowButtonHover: function onOverflowButtonHover(args)
{ }
 }); | Not applicable |

| Fires when the reminder action takes place | **Event:** *reminder*

\$("#Schedule").ejSchedule({
 reminder: function onReminder(args) {}
 }); | Not applicable
|

| Fires on resizing event | **Event:** *resize*

 \$("#Schedule").ejSchedule({
 resize: function
onResize(args) {}
 }); | **Event:** *resize*

 var schObj = new ej.schedule.Schedule({

resize: function onResize(args) {}
 }); |

| Fires on resize start | **Event:** *resizeStart*

 `$("#Schedule").ejSchedule({
resizeStart: function onResizeStart(args) {}
 });` | **Event:** *resizeStart*

 `var schObj = new ej.schedule.Schedule({
resizeStart: function onResizeStart (args) {}
 });` |

| Fires on resize stop | **Event:** *resizeStop*

 `$("#Schedule").ejSchedule({
resizeStop: function onResizeStop(args) {}
 });` | **Event:** *resizeStop*

 `var schObj = new ej.schedule.Schedule({
resizeStop: function onResizeStop (args) {}
 });` |

| Fires on rendering of every scheduler elements | **Event:** *queryCellInfo*

 `$("#Schedule").ejSchedule({
queryCellInfo: function onQueryCellInfo (args) {}
 | Event: renderCell

 var schObj = new ej.schedule.Schedule({
renderCell: function onRenderCell (args) {}
 });` |

| Fires before the event rendering on UI | Not applicable | **Event:** *eventRendered*

 `var schObj = new ej.schedule.Schedule({
eventRendered: function onEventRendered (args) {}
 });` |

You can refer to our [JavaScript Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Scheduler example](#) to know how to present and manipulate data.

How To

Add edit and remove events in EJ2 JavaScript Schedule control

CRUD actions can be manually performed on appointments using `addEvent`, `saveEvent` and `deleteEvent` methods as shown below.

Normal event

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleData: Object[] = [{
  Id: 3,
  Subject: 'Testing',
  StartTime: new Date(2018, 1, 11, 9, 0),
  EndTime: new Date(2018, 1, 11, 10, 0),
  IsAllDay: false
}, {
  Id: 4,
  Subject: 'Vacation',
  StartTime: new Date(2018, 1, 13, 9, 0),
  EndTime: new Date(2018, 1, 13, 10, 0),
  IsAllDay: false
}];
let scheduleObj: Schedule = new Schedule({
  height: '550px',
  selectedDate: new Date(2018, 1, 15),
  views: ['Day', 'Week', 'WorkWeek', 'Month'],
  eventSettings: {
    dataSource: scheduleData
  }
});
```

```

scheduleObj.appendTo('#Schedule');
let button: Button = new Button();
button.appendTo('#btn1');
button.element.onclick = (): void => {
let Data: Object[] = [{
    Id: 1,
    Subject: 'Conference',
    StartTime: new Date(2018, 1, 12, 9, 0),
    EndTime: new Date(2018, 1, 12, 10, 0),
    IsAllDay: false
}],{
    Id: 2,
    Subject: 'Meeting',
    StartTime: new Date(2018, 1, 15, 10, 0),
    EndTime: new Date(2018, 1, 15, 11, 30),
    IsAllDay: false
}
});
scheduleObj.addEvent(Data);
};
let button2: Button = new Button();
button2.appendTo('#btn2');
button2.element.onclick = (): void => {
let Data: Object = {
    Id: 3,
    Subject: 'Testing-edited',
    StartTime: new Date(2018, 1, 11, 10, 0),
    EndTime: new Date(2018, 1, 11, 11, 0),
    IsAllDay: false
};
scheduleObj.saveEvent(Data);
};
let button3: Button = new Button();
button3.appendTo('#btn3');
button3.element.onclick = (): void => {
scheduleObj.deleteEvent(4);
};
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Schedule Typescript Component</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="btn1">Add</button>
        <button id="btn2">Edit</button>
        <button id="btn3">Delete</button>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Recurrence event

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
schedule';
import { Button } from '@syncfusion/ej2-buttons';
import { DataManager, Query, Predicate } from '@syncfusion/ej2-data';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleData: Object[] = [{
    Id: 3,
    Subject: 'Testing',
    StartTime: new Date(2018, 1, 11, 9, 0),
    EndTime: new Date(2018, 1, 11, 10, 0),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3'
}, {
    Id: 4,
    Subject: 'Vacation',
    StartTime: new Date(2018, 1, 12, 11, 0),
    EndTime: new Date(2018, 1, 12, 12, 0),
    IsAllDay: false,
    RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
}];
let scheduleObj: Schedule = new Schedule({
    height: '550px',

```

```

        selectedDate: new Date(2018, 1, 15),
        views: ['Day', 'Week', 'WorkWeek', 'Month'],
        eventSettings: {
            dataSource: scheduleData
        }
    });
    scheduleObj.appendTo('#Schedule');
    let button: Button = new Button();
    button.appendTo('#btn1');
    button.element.onclick = (): void => {
    let Data: Object[] = [{
        Id: 1,
        Subject: 'Conference',
        StartTime: new Date(2018, 1, 15, 9, 0),
        EndTime: new Date(2018, 1, 15, 10, 0),
        IsAllDay: false,
        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
    }];
    scheduleObj.addEvent(Data);
    button.element.setAttribute('disabled', 'true');
    };
    let button2: Button = new Button();
    button2.appendTo('#btn2');
    button2.element.onclick = (): void => {
    let data: Object = new
    DataManager(scheduleObj.getCurrentViewEvents()).executeLocal(new
    Query().where('RecurrenceID', 'equal', 3));
    data[0].Subject = 'Occurrence edited';
    scheduleObj.saveEvent(data[0], 'EditOccurrence');
    button2.element.setAttribute('disabled', 'true');
    };
    let button3: Button = new Button();
    button3.appendTo('#btn3');
    button3.element.onclick = (): void => {
    let scheduleData: { [key: string]: Object }[] = [{
        Id: 4,
        Subject: 'Vacation',
        RecurrenceID: 4,
        StartTime: new Date(2018, 1, 12, 11, 0),
        EndTime: new Date(2018, 1, 12, 12, 0),
        IsAllDay: false,
        RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
    }];
    scheduleObj.deleteEvent(scheduleData, 'DeleteSeries');
    button3.element.setAttribute('disabled', 'true');
    };

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="btn1">Add</button>
        <button id="btn2">Edit</button>
        <button id="btn3">Delete</button>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set default value for event fields in EJ2 JavaScript Schedule control

Event window default fields name like Title, Location, etc.. can be customized and default value can be set to Subject field using **default** property which will be added if an appointment is created with empty subject.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '500px',
    selectedDate: new Date(2018, 1, 15),

```

```

    eventSettings: {
        dataSource: scheduleData,
        fields: {
            subject: { title: 'Event Name', name: 'Subject', default: 'Add
Name' },
            location: { title: 'Event Location', name: 'Location', default:
'USA' },
            description: { title: 'Summary', name: 'Description' },
            startTime: { title: 'From', name: 'StartTime' },
            endTime: { title: 'To', name: 'EndTime' }
        }
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <style>
        .e-textlabel {
            font-weight: bold;
            padding-right: 5px;
        }
        .custom-event-editor td {
            padding: 5px;
        }
        .e-quick-popup-wrapper .e-cell-content {
            padding: 0 10px 10px 10px;
        }
        .e-quick-popup-wrapper .e-cell-content div {
            padding-bottom: 10px;
        }
    </style>

```



```

.e-quick-popup-wrapper .e-cell-content .e-field {
    border-left-width: 0;
    border-top-width: 0;
    border-right-width: 0;
    height: 25px;
    width: 100%;
}
.e-quick-popup-wrapper .e-event-content {
    display: flex;
}
.e-quick-popup-wrapper .e-event-header {
    position: absolute;
    right: 0;
}
.e-quick-popup-wrapper .e-cell-footer .e-event-create,
.e-quick-popup-wrapper .e-event-footer .e-event-edit {
    position: absolute;
    right: 0;
}
.e-quick-popup-wrapper .e-event-footer .e-event-delete {
    padding-left: 100px;
}
.e-quick-popup-wrapper .e-event-content .subject {
    padding-top: 10px;
    font-weight: 500;
    font-size: 14px;
}
</style>
<script id="headerTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-header">
            <div class="e-header-icon-wrapper">
                <button class="e-close" title="Close"></button>
            </div>
        </div>
        ${else}
        <div class="e-event-header">
            <div class="e-header-icon-wrapper">
                <button class="e-close" title="CLOSE"></button>
            </div>
        </div>
        ${/if}
    </div>
</script>
<script id="contentTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-content">
            <form class="e-schedule-form">
                <div>
                    <input class="subject e-field" type="text"
name="Subject" placeholder="Title">
                </div>
                <div>
                    <input class="location e-field" type="text"
name="Location" placeholder="Location">

```

```

        </div>
    </form>
</div>
    <{else}>
    <div class="e-event-content">
        <div class="e-subject-wrap">
            <{if (Subject)}>
            <div class="subject">${Subject}</div></if> <{if (City)}>
            <div class="location">${City}</div></if> <{if
(Description)}>
            <div class="description">${Description}</div></if>
        </div>
    </div>
    <{/if>
</div>
</script>
<script id="footerTemplate" type="text/x-template">
    <div>
        <{if (elementType === 'cell')}>
        <div class="e-cell-footer">
            <button class="e-event-details" title="Additional
Details">Additional Details</button>
            <button class="e-event-create" title="Add">Add</button>
        </div>
        <{else}>
        <div class="e-event-footer">
            <button class="e-event-edit" title="Edit">Edit</button>
            <button class="e-event-delete"
title="Delete">Delete</button>
        </div>
        <{/if>
    </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open event editor manually in EJ2 JavaScript Schedule control

Open Editor Window externally

Scheduler allows the user to manually open the event editor on specific time or on certain events using **openEditor** method. To open the editor on specific range of time, user need to pass the cell details as first argument and **Add** as second argument whereas to open it on event pass that event detail and **Save** as arguments. In the following code example, on clicking the respective button will open the respective editor window manually.

INDEX.TS

```
import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let button1: Button = new Button();
button1.appendTo('#btn1');
let button2: Button = new Button();
button2.appendTo('#btn2');
button1.element.onclick = (): void => {
    let cellData: Object = {
        startTime: new Date(2018, 1, 15, 10, 0),
        endTime: new Date(2018, 1, 15, 11, 0),
    };
    scheduleObj.openEditor(cellData, 'Add');
};
button2.element.onclick = (): void => {
    let eventData: Object = {
        Id: 4,
        Subject: 'Meteor Showers in 2018',
        StartTime: new Date(2018, 1, 14, 13, 0),
        EndTime: new Date(2018, 1, 14, 14, 30)
    };
    scheduleObj.openEditor(eventData, 'Save');
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-lg-12 property-section">
        <button id="btn1">Open editor on cell</button>
        <button id="btn2">Open editor on event</button>
    </div>
    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open editor window on single click

By default, Scheduler Editor window will open when double clicking the cells or appointments. You can also open the editor window with single click by using `openEditor` method in `eventClick` and `cellClick` events of scheduler and setting `false` to `showQuickInfo`. The following example shows how to open editor window on single click of cells and appointments.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, CellClickEventArgs,
EventClickArgs } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
import { schedulerData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',

```

```

selectedDate: new Date(2021, 7, 15),
views: ['Day', 'Week', 'WorkWeek', 'Month'],
eventSettings: { dataSource: schedulerData },
showQuickInfo: false,
cellClick: (args: CellClickEventArgs) => {
    scheduleObj.openEditor(args, 'Add');
},
eventClick: (args: EventClickArgs) => {
    if (!args.event as any).RecurrenceRule {
        scheduleObj.openEditor(args.event, 'Save');
    }
    else {
        scheduleObj.quickPopup.openRecurrenceAlert();
    }
}
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent date navigation in EJ2 JavaScript Schedule control

We can prevent navigation while clicking on the date header by simply removing `e-navigate` class from header cells which can be achieved in the `renderCell` event as shown in the following code example.

INDEX.TS

```

import { removeClass } from '@syncfusion/ej2-base';
import { Schedule, Day, Week, WorkWeek, Month, Agenda, RenderCellEventArgs }
from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '500px',
    selectedDate: new Date(2018, 1, 15),
    currentView: 'WorkWeek',
    renderCell: (args: RenderCellEventArgs) => {
        if(args.elementType === "dateHeader" || args.elementType ===
"monthCells") {
            removeClass(args.element.childNodes, "e-navigate");
        }
    },
    eventSettings: {
        dataSource: scheduleData
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-textlabel {
        font-weight: bold;
        padding-right: 5px;
    }
    .custom-event-editor td {
        padding: 5px;
    }
    .e-quick-popup-wrapper .e-cell-content {
        padding: 0 10px 10px 10px;
    }
    .e-quick-popup-wrapper .e-cell-content div {
        padding-bottom: 10px;
    }
    .e-quick-popup-wrapper .e-cell-content .e-field {
        border-left-width: 0;
        border-top-width: 0;
        border-right-width: 0;
        height: 25px;
        width: 100%;
    }
    .e-quick-popup-wrapper .e-event-content {
        display: flex;
    }
    .e-quick-popup-wrapper .e-event-header {
        position: absolute;
        right: 0;
    }
    .e-quick-popup-wrapper .e-cell-footer .e-event-create,
    .e-quick-popup-wrapper .e-event-footer .e-event-edit {
        position: absolute;
        right: 0;
    }
    .e-quick-popup-wrapper .e-event-footer .e-event-delete {
        padding-left: 100px;
    }
    .e-quick-popup-wrapper .e-event-content .subject {
        padding-top: 10px;
        font-weight: 500;
        font-size: 14px;
    }
</style>
<script id="headerTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-header">
            <div class="e-header-icon-wrapper">

```

```

        <button class="e-close" title="Close"></button>
    </div>
</div>
${else}
<div class="e-event-header">
    <div class="e-header-icon-wrapper">
        <button class="e-close" title="CLOSE"></button>
    </div>
</div>
${/if}
</div>
</script>
<script id="contentTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-content">
            <form class="e-schedule-form">
                <div>
                    <input class="subject e-field" type="text"
name="Subject" placeholder="Title">
                </div>
                <div>
                    <input class="location e-field" type="text"
name="Location" placeholder="Location">
                </div>
            </form>
        </div>
        ${else}
        <div class="e-event-content">
            <div class="e-subject-wrap">
                ${if (Subject)}
                <div class="subject">${Subject}</div>${/if} ${if (City)}
                <div class="location">${City}</div>${/if} ${if
(Description)}
                <div class="description">${Description}</div>${/if}
            </div>
        </div>
        ${/if}
    </div>
</script>
<script id="footerTemplate" type="text/x-template">
    <div>
        ${if (elementType === 'cell')}
        <div class="e-cell-footer">
            <button class="e-event-details" title="Additional
Details">Additional Details</button>
            <button class="e-event-create" title="Add">Add</button>
        </div>
        ${else}
        <div class="e-event-footer">
            <button class="e-event-edit" title="Edit">Edit</button>
            <button class="e-event-delete"
title="Delete">Delete</button>
        </div>
        ${/if}
    </div>
</script>

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Half yearly view in EJ2 JavaScript Schedule control

The year view of our scheduler displays all the 365 days and their related appointments of a particular year. You can customize the year view by using the following properties.

- [firstMonthOfYear](#)
- [monthsCount](#)
- [monthHeaderTemplate](#)

In the following code example, you can see how to render only the last six months of a year in the scheduler. To start with the month of June, `firstMonthYear` is set to 6 and `monthsCount` is set to 6 to render only 6 months.

INDEX.TS

```

import { Schedule, Year, TimelineYear, DragAndDrop, Resize } from
'@syncfusion/ej2-schedule';
import { Internationalization } from '@syncfusion/ej2-base';
import { resourceData } from './datasource.ts';
Schedule.Inject(Year, TimelineYear, DragAndDrop, Resize);
interface TemplateFunction extends Window {
    getMonthHeaderText?: Function;
}
(window as TemplateFunction).getMonthHeaderText = (date: Date) => { return
date.toLocaleString("en-us", { month: "long" }) + " " + date.getFullYear();
};
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '555px',
    selectedDate: new Date(2021, 7, 4),
    firstMonthOfYear: 6,
    monthsCount: 6,
    monthHeaderTemplate: '<div class="date-
text">${getMonthHeaderText(data.date)}</div>',
    resourceHeaderTemplate: '#resourceTemplate',

```

```

views: [
    { option: 'Year'},
    { option: 'TimelineYear', displayName: 'Horizontal Year',
isSelected: true },
    { option: 'TimelineYear', displayName: 'Vertical Year', orientation:
'Vertical' }
],
group: {
    resources: ['Projects', 'Categories']
},
resources: [
    {
        field: 'ProjectId', title: 'Choose Project', name: 'Projects',
        dataSource: [
            { text: 'PROJECT 1', id: 1, color: '#cb6bb2' },
            { text: 'PROJECT 2', id: 2, color: '#56ca85' },
            { text: 'PROJECT 3', id: 3, color: '#df5286' }
        ],
        textField: 'text', idField: 'id', colorField: 'color'
    }, {
        field: 'TaskId', title: 'Category',
        name: 'Categories', allowMultiple: true,
        dataSource: [
            { text: 'Nancy', id: 1, groupId: 1, color: '#df5286' },
            { text: 'Steven', id: 2, groupId: 2, color: '#7fa900' },
            { text: 'Robert', id: 3, groupId: 3, color: '#ea7a57' },
            { text: 'Smith', id: 4, groupId: 1, color: '#5978ee' },
            { text: 'Micheal', id: 5, groupId: 2, color: '#df5286' },
            { text: 'Root', id: 6, groupId: 3, color: '#00bdae' }
        ],
        textField: 'text', idField: 'id', groupIDField: 'groupId',
        colorField: 'color'
    }
],
eventSettings: { dataSource: resourceData }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<style>
    .e-schedule .e-vertical-view .e-resource-cells {
        height: 62px;
    }

    .e-schedule .template-wrap {
        display: flex;
        text-align: left;
    }

    .e-schedule .template-wrap .resource-details {
        padding-left: 10px;
    }

    .e-schedule .template-wrap .resource-details .resource-name {
        font-size: 14px;
        font-weight: 500;
        margin-top: 5px;
    }

    .e-schedule.e-device .template-wrap .resource-details .resource-name
{
        font-size: inherit;
        font-weight: inherit;
    }

    .e-schedule.e-device .e-resource-tree-popup .e-fullrow {
        height: 50px;
    }

</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <script id="resourceTemplate" type="text/x-template">
        <div class='template-wrap'>
            <div class="resource-details">
                <div class="resource-name">${resourceData.text}</div>
            </div>
        </div>
    </script>
    <div id="container">

```

```

        <div id="Schedule"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set different work hours in EJ2 JavaScript Schedule control

By default, the work hours of the Scheduler is highlighted based on the start and end values provided within the `workHours` property which remains same for all days. To highlight different work hours range for different days, `setWorkHours` method. You can pass date object/ multiple date objects collection as first argument and start and end time need to be added as work hours should be passed as second and third arguments respectively. In the following code example, on button click 11:00 AM to 08:00 PM of 15th and 17th February has been added in work hours.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-schedule';
import { Button } from '@syncfusion/ej2-buttons';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month);
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2018, 1, 15),
    workHours: {
        highlight: true,
        start: '09:00',
        end: '11:00'
    },
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
let button: Button = new Button();
button.appendTo('#btn1');
let btnElement: HTMLElement =
<HTMLElement>document.querySelector('#btn1');
let btnElement2: HTMLElement =
<HTMLElement>document.querySelector('#btn2');
btnElement.innerText = 'Change the work hours';
btnElement2.style.display='none';
button.element.onclick = (): void => {
    let dates: Date[] = [new Date(2018, 1, 15), new Date(2018, 1, 17)];
    scheduleObj.setWorkHours(dates, '11:00', '20:00');
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Schedule Typescript Component</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Schedule Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div class="col-lg-12 property-section">
        <button id="btn1">Open editor on cell</button>
        <button id="btn2">Open editor on event</button>
    </div>
    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Zoom in and zoom out the schedule in EJ2 JavaScript Schedule control

By default Scheduler component doesn't have the Zoom in/out support. Using the `timeScale` and `headerRows` properties of our scheduler, we can achieve this.

Refer to the following code example.

INDEX.TS

```

import { Schedule, Week, Month, Agenda, TimelineViews, TimelineMonth,
NavigatingEventArgs } from '@syncfusion/ej2-schedule';
import { scheduleData } from './datasource.ts';
Schedule.Inject(Month, Week, Agenda, TimelineViews, TimelineMonth);
let scheduleObj: Schedule = new Schedule({
    height: "600px",
    width: "100%",
    selectedDate: new Date(2020, 2, 12),
    currentView: "TimelineWeek",
    views: ["TimelineDay", "TimelineWeek", "TimelineMonth"],
    timeScale: {
        enable: true,
        interval: 60,
        slotCount: 2
    },
    navigating: (args: NavigatingEventArgs) => {
        if (args.currentView !== "TimelineMonth") {
            scheduleObj.headerRows = [];
        }
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo('#Schedule');
window.addEventListener("wheel", event => {
    const delta = Math.sign(event.deltaY);
    if (scheduleObj.currentView === "TimelineMonth") {
        var len = scheduleObj.headerRows.length;
        if (delta < 0) {
            if (len == 0) {
                scheduleObj.setProperties({
                    headerRows: [{ option: "Date" }]
                });
            } else if (len == 1) {
                scheduleObj.setProperties({
                    headerRows: [{ option: "Week" }, { option: "Date" }]
                });
            } else if (len == 2) {
                scheduleObj.setProperties({
                    headerRows: [
                        { option: "Month" },
                        { option: "Week" },
                        { option: "Date" }
                    ]
                });
            } else if (len == 3) {
                scheduleObj.setProperties({
                    headerRows: [
                        { option: "Year" },
                        { option: "Month" },
                        { option: "Week" },
                        { option: "Date" }
                    ]
                });
            }
        } else if (delta > 0) {
            if (len == 2) {
                scheduleObj.setProperties({

```

```

        headerRows: [{ option: "Date" }]
    });
    } else if (len == 3) {
        scheduleObj.setProperties({
            headerRows: [{ option: "Week" }, { option: "Date" }]
        });
    } else if (len == 4) {
        scheduleObj.setProperties({
            headerRows: [
                { option: "Month" },
                { option: "Week" },
                { option: "Date" }
            ]
        });
    } else if (len == 5) {
        scheduleObj.setProperties({
            headerRows: [
                { option: "Year" },
                { option: "Month" },
                { option: "Week" },
                { option: "Date" }
            ]
        });
    }
} else {
    if (delta > 0 && scheduleObj.timeScale.slotCount > 1) {
        scheduleObj.timeScale.slotCount -= 1;
    } else if (delta < 0 && scheduleObj.timeScale.slotCount <= 8) {
        scheduleObj.timeScale.slotCount += 1;
    }
}
});

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Show entire time in responsive mode in EJ2 JavaScript Schedule control

9 AM is visible since it has enough space in responsive mode, but if you set 08:45 AM as start hour of scheduler then the time slots will not show full time in the time slot of responsive scheduler, for which we can use `majorSlotTemplate` to set proper time slots in scheduler.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek } from "@syncfusion/ej2-schedule";
import { Internationalization } from "@syncfusion/ej2-base";
import { scheduleData } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek);
let instance: Internationalization = new Internationalization();
(window as TemplateFunction).majorSlotTemplate = (date: Date) => {
    return instance.formatDate(date, { skeleton: "hm" });
};
interface TemplateFunction extends Window {
    majorSlotTemplate?: Function;
}
let scheduleObj: Schedule = new Schedule({
    width: "100%",
    height: "550px",
    selectedDate: new Date(2018, 1, 15),
    startHour: "08:45",
    views: ["Day", "Week", "WorkWeek"],
    timeScale: {
        majorSlotTemplate: "#majorSlotTemplate"
    },
    eventSettings: { dataSource: scheduleData }
});
scheduleObj.appendTo("#Schedule");

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script id="majorSlotTemplate" type="text/x-template">
    <div>${majorSlotTemplate(data.date)}</div>
  </script><script id="minorSlotTemplate" type="text/x-template">
    <div style="text-align: right; margin-right:
15px">${minorSlotTemplate(data.date)}</div>
  </script><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Quick info template in EJ2 JavaScript Schedule control

This demo showcases the quick popups for cells and appointments with the customized templates.

INDEX.TS

```
import { extend, Internationalization } from '@syncfusion/ej2-base';
import { Button } from '@syncfusion/ej2-buttons';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { TextBox } from '@syncfusion/ej2-inputs';
import {
    Schedule, Day, Week, WorkWeek, Month, Agenda, EventRenderedArgs, Resize,
    DragAndDrop,
    ResourcesModel, PopupOpenEventArgs, EJ2Instance, CellClickEventArgs,
    CurrentAction
} from '@syncfusion/ej2-schedule';
import { scheduleData } from '../datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda, Resize, DragAndDrop);

interface TemplateFunction extends Window {
    getHeaderDetails?: Function;
    getHeaderStyles?: Function;
    getEventType?: Function;
    getResourceData?: Function;
}

(window as TemplateFunction).getResourceData = (data: { [key: string]:
Object }) => {
    let resources: ResourcesModel =
scheduleObj.getResourceCollections().slice(-1)[0];
    let resourceData: { [key: string]: Object } = (resources.dataSource
as Object[]).filter((resource: { [key: string]: Object }) =>
    resource.Id === data.RoomId)[0] as { [key: string]: Object };
    return resourceData;
};

(window as TemplateFunction).getHeaderDetails = (data: { [key: string]:
Date }) => {
    let intl: Internationalization = new Internationalization();
    return intl.formatDate(data.StartTime, { type: 'date', skeleton:
'full' }) + ' (' +
        intl.formatDate(data.StartTime, { skeleton: 'hm' }) + ' - ' +
intl.formatDate(data.EndTime, { skeleton: 'hm' }) + ')';
};

(window as TemplateFunction).getHeaderStyles = (data: { [key: string]:
Object }) => {
    if (data.elementType === 'cell') {
        return 'align-items: center; color: #919191;';
    } else {
        let resourceData: { [key: string]: Object } = (window as
TemplateFunction).getResourceData(data);
        return 'background: ' + resourceData.Color + '; color: #FFFFFF;';
    }
};

let buttonClickActions: Function = (e: Event) => {
    let quickPopup: HTMLElement = scheduleObj.element.querySelector('.e-
quick-popup-wrapper') as HTMLElement;
    let getSlotData: Function = (): { [key: string]: Object } => {
        let cellDetails: CellClickEventArgs =
scheduleObj.getCellDetails(scheduleObj.getSelectedElements());
```

```

        let addObj: { [key: string]: Object } = {};
        addObj.Id = scheduleObj.getEventMaxID();
        addObj.Subject = ((quickPopup.querySelector('#title') as
EJ2Instance).ej2_instances[0] as TextBox).value;
        addObj.StartTime = new Date(+cellDetails.startTime);
        addObj.EndTime = new Date(+cellDetails.endTime);
        addObj.Description = ((quickPopup.querySelector('#notes') as
EJ2Instance).ej2_instances[0] as TextBox).value;
        addObj.RoomId = ((quickPopup.querySelector('#eventType') as
EJ2Instance).ej2_instances[0] as DropDownList).value;
        return addObj;
    };
    if ((e.target as HTMLElement).id === 'add') {
        let addObj: { [key: string]: Object } = getSlotData();
        scheduleObj.addEvent(addObj);
    } else if ((e.target as HTMLElement).id === 'delete') {
        let eventDetails: { [key: string]: Object } =
scheduleObj.activeEventData.event as { [key: string]: Object };
        let currentAction: CurrentAction;
        if (eventDetails.RecurrenceRule) {
            currentAction = 'DeleteOccurrence';
        }
        scheduleObj.deleteEvent(eventDetails, currentAction);
    } else {
        let isCellPopup: boolean =
quickPopup.firstElementChild.classList.contains('e-cell-popup');
        let eventDetails: { [key: string]: Object } = isCellPopup ?
getSlotData() :
            scheduleObj.activeEventData.event as { [key: string]: Object
};
        let currentAction: CurrentAction = isCellPopup ? 'Add' : 'Save';
        if (eventDetails.RecurrenceRule) {
            currentAction = 'EditOccurrence';
        }
        scheduleObj.openEditor(eventDetails, currentAction, true);
    }
    scheduleObj.closeQuickInfoPopup();
};
(window as TemplateFunction).getEventType = (data: { [key: string]: Date
}) => {
    let resourceData: { [key: string]: Object } = (window as
TemplateFunction).getResourceData(data);
    return resourceData.Name;
};
let data: Object[] = <Object[]>extend([], scheduleData, null, true);
let scheduleObj: Schedule = new Schedule({
    width: '100%',
    height: '650px',
    selectedDate: new Date(2020, 0, 9),
    eventSettings: { dataSource: data },
    resources: [{
        field: 'RoomId', title: 'Room Type', name: 'MeetingRoom',
textField: 'Name', idField: 'Id', colorField: 'Color',
        dataSource: [
            { Name: 'Jammy', Id: 1, Capacity: 20, Color: '#ea7a57',
Type: 'Conference' },

```

```

        { Name: 'Tweety', Id: 2, Capacity: 7, Color: '#7fa900',
Type: 'Cabin' },
        { Name: 'Nestle', Id: 3, Capacity: 5, Color: '#5978ee',
Type: 'Cabin' },
        { Name: 'Phoenix', Id: 4, Capacity: 15, Color: '#fec200',
Type: 'Conference' },
        { Name: 'Mission', Id: 5, Capacity: 25, Color: '#df5286',
Type: 'Conference' },
        { Name: 'Hangout', Id: 6, Capacity: 10, Color: '#00bdae',
Type: 'Cabin' },
        { Name: 'Rick Roll', Id: 7, Capacity: 20, Color: '#865fcf',
Type: 'Conference' },
        { Name: 'Rainbow', Id: 8, Capacity: 8, Color: '#1aaa55',
Type: 'Cabin' },
        { Name: 'Swarm', Id: 9, Capacity: 30, Color: '#df5286',
Type: 'Conference' },
        { Name: 'Photogenic', Id: 10, Capacity: 25, Color:
'#710193', Type: 'Conference' }
    ]
  },
  quickInfoTemplates: {
    header: '#header-template',
    content: '#content-template',
    footer: '#footer-template'
  },
  eventRendered: (args: EventRenderedArgs) => {
    let categoryColor: string = args.data.CategoryColor as string;
    if (!args.element || !categoryColor) {
      return;
    }
    if (scheduleObj.currentView === 'Agenda') {
      (args.element.firstChild as
HTMLElement).style.borderLeftColor = categoryColor;
    } else {
      args.element.style.backgroundColor = categoryColor;
    }
  },
  popupOpen: (args: PopupOpenEventArgs) => {
    if (args.type === 'QuickInfo') {
      let titleObj: TextBox = new TextBox({ placeholder: 'Title'
});
      titleObj.appendTo(args.element.querySelector('#title') as
HTMLElement);
      let typeObj: DropDownList = new DropDownList({
        dataSource: scheduleObj.getResourceCollections().slice(-
1)[0].dataSource as { [key: string]: Object }[],
        placeholder: 'Choose Type',
        fields: { text: 'Name', value: 'Id' },
        index: 0
      });
      typeObj.appendTo(args.element.querySelector('#eventType') as
HTMLElement);
      let notesObj: TextBox = new TextBox({ placeholder: 'Notes'
});
      notesObj.appendTo(args.element.querySelector('#notes') as
HTMLElement);
    }
  }
}

```

```

        let moreDetailsBtn: HTMLButtonElement =
args.element.querySelector('#more-details') as HTMLButtonElement;
        if (moreDetailsBtn) {
            let moreObj: Button = new Button({
                content: 'More Details', cssClass: 'e-flat',
                isPrimary:
args.element.firstElementChild.classList.contains('e-event-popup')
            });
            moreObj.appendTo(moreDetailsBtn);
            moreDetailsBtn.onclick = (e: Event) => {
buttonClickActions(e); };
        }
        let addBtn: HTMLButtonElement =
args.element.querySelector('#add') as HTMLButtonElement;
        if (addBtn) {
            new Button({ content: 'Add', cssClass: 'e-flat',
isPrimary: true }, addBtn);
            addBtn.onclick = (e: Event) => { buttonClickActions(e);
};
        }
        let deleteBtn: HTMLButtonElement =
args.element.querySelector('#delete') as HTMLButtonElement;
        if (deleteBtn) {
            new Button({ content: 'Delete', cssClass: 'e-flat' },
deleteBtn);
            deleteBtn.onclick = (e: Event) => {
buttonClickActions(e); };
        }
    }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div class="control-section">
    <div class="content-wrapper">
      <div id="Schedule"></div>
    </div>
  </div>

  <script id="header-template" type="text/x-template">
    <div class="quick-info-header">
      <div class="quick-info-header-content"
style="${getHeaderStyles(data)}">
        <div class="quick-info-title">${if (elementType ==
"cell")}Add Appointment${else}Appointment Details${if}</div>
        <div class="duration-text">${getHeaderDetails(data)}</div>
      </div>
    </div>
  </script>

  <script id="content-template" type="text/x-template">
    <div class="quick-info-content">
      ${if (elementType == "cell")}
      <div class="e-cell-content">
        <div class="content-area">
          <input id="title" placeholder="Title" />
        </div>
        <div class="content-area">
          <input id="eventType" placeholder="Choose Type" />
        </div>
        <div class="content-area">
          <input id="notes" placeholder="Notes" />
        </div>
      </div>
      ${else}
      <div class="event-content">
        <div class="meeting-type-wrap">
          <label>Subject</label>:
          <span>${Subject}</span>
        </div>
        <div class="meeting-subject-wrap">
          <label>Type</label>:
          <span>${getEventType(data)}</span>
        </div>
        <div class="notes-wrap">
          <label>Notes</label>:
          <span>${Description}</span>
        </div>
      </div>
    </div>
  </script>

```

```

        </div>
    </div>
    ${/if}
</div>
</script>

<script id="footer-template" type="text/x-template">
    <div class="quick-info-footer">
        ${if (elementType == "cell")}
        <div class="cell-footer">
            <button id="more-details">More Details</button>
            <button id="add">Add</button>
        </div>
        ${else}
        <div class="event-footer">
            <button id="delete">Delete</button>
            <button id="more-details">More Details</button>
        </div>
        ${/if}
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable scroll option on all day section in EJ2 JavaScript Schedule control

When you have larger number of appointments in all-day row, it is difficult to view all the appointments properly. In that case you can enable scroller option for all-day row by setting true to `enableAllDayScroll` whereas its default value is false. When setting this property to true, individual scroller for all-day row is enabled when it reaches its maximum height on expanding.

Note: This property is not applicable for Scheduler with height `auto`.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month, Agenda } from
'@syncfusion/ej2-schedule';
import { generateObject } from './datasource.ts';
Schedule.Inject(Day, Week, WorkWeek, Month, Agenda);
let data: Object[] = generateObject();
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2021, 3, 28),
    enableAllDayScroll: true,
    eventSettings: { dataSource: data }
});
scheduleObj.appendTo('#Schedule');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Schedule Typescript Component</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Schedule Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Schedule"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Manual refresh in EJ2 JavaScript Schedule control

In Scheduler, we can able to refresh the layout manually without re-render the DOM element by using the [refreshLayout](#) public method. The following example code explains to know how to use the refreshLayout method.

INDEX.TS

```

import { Schedule, Day, Week, WorkWeek, Month } from '@syncfusion/ej2-
schedule';
import { Button } from '@syncfusion/ej2-buttons';
Schedule.Inject(Day, Week, WorkWeek, Month);

```



```

let scheduleData: Object[] = [
    {
        Id: 1,
        Subject: 'Testing',
        StartTime: new Date(2021, 10, 16, 10, 0),
        EndTime: new Date(2021, 10, 16, 12, 0),
        IsAllDay: false
    }, {
        Id: 2,
        Subject: 'Vacation',
        StartTime: new Date(2021, 10, 18, 10, 0),
        EndTime: new Date(2021, 10, 18, 12, 0),
        IsAllDay: false
    }
];
let scheduleObj: Schedule = new Schedule({
    height: '550px',
    selectedDate: new Date(2021, 10, 15),
    views: ['Day', 'Week', 'WorkWeek', 'Month'],
    eventSettings: {
        dataSource: scheduleData
    }
});
scheduleObj.appendTo('#Schedule');
let button: Button = new Button({
    cssClass: 'e-info'
});
button.appendTo('#btn1');
button.element.onclick = (): void => {
    scheduleObj.refreshLayout();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Schedule Typescript Component</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Schedule Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
schedule/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="btn1">Refresh Layout</button>
        <div id="Schedule"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sidebar

Getting started in EJ2 JavaScript Sidebar control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization

Create an app folder **myapp** in local machine to initialize Essential JS 2 JavaScript components.

Using either of the following way to refer the required script and styles.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder **myapp** for Essential JS 2 JavaScript components.

Step 2: You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

Syntax:

Script: **(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\dist\global\{PACKAGE_NAME}.min.js**

Styles: **(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\styles\material.css**

Example:

Script: C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-navigations\dist\global\ej2-navigations.min.js

Styles: C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-navigations\styles\material.css

The below located script and style file contains all Syncfusion JavaScript (ES5) UI control resources in a single file.

Scripts: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\dist\ej2.min.js

Styles: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\material.css

Step 3: Create a folder `myapp/resources` and copy/paste the above mentioned packages from the above installed location to `myapp/resources` location.

Step 4: Create a HTML page (`index.html`) in `myapp` location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Sidebar's global and dependent script -->
<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the `div` element and initiate the `Essential JS 2 Sidebar` component in the `index.html` by using following code

```
`html
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Sidebar's global and dependent script -->
<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element -->
<div id='container'>
<aside id="default">
<div class="title"> Sidebar </div>
</aside>
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
<div class="title">Main content</div>
<div class="sub-title"> Content goes here</div>
</div>
</div>
<script>
// initialize Sidebar component
var defaultSidebar = new ej.navigations.Sidebar();
// Render initialized sidebar.
defaultSidebar.appendTo('#default')
</script>
</body>
</html>
,
```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Sidebar** component.

Using CDN link for script and style reference

Step 1: The Essential JS 2 components scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Style: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css>

Step 2: Have to add CDN global script and style for Sidebar and its dependent packages in `myapp/index.html` like below.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Sidebar Component</title>
<!-- Sidebar and its dependent theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css" />
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css" />
<!-- Essential JS 2 Sidebar's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element -->
<div id='container'>
<aside id="default">
<div class="title"> Sidebar </div>
</aside>
<!-- end of sidebar element -->
```

```

<!-- main content declaration -->
<div>
<div class="title">Main content</div>
<div class="sub-title"> Content goes here</div>
</div>
</div>
<script>
// initialize Sidebar component
var defaultSidebar = new ej.navigations.Sidebar();
// Render initialized sidebar.
defaultSidebar.appendTo('#default')
</script>
</body>
</html>
`

```

When referencing CDN links in application, always ensure the network connection will be in enabled state.

Enable backdrop

Enabling the `showBackdrop` in the Sidebar component will prevent the main content from user interactions, when it is in expanded state.

Here, the DOM elements will not get changed. It only closes the main content by covering with a black backdrop overlay and focuses the Sidebar in the screen. Sidebar can be rendered with specific width by setting `width` property.

Note: To achieve a proper **backdrop**, we suggest that you create a wrapper parent container for the div block in which you intend to enable the backdrop. Set the class name of this parent container as the **target** for the Sidebar. Alternatively, you can place an empty div container after the target container.

INDEX.JS

```

ej.base.enableRipple(true);
//sidebar initialization
var defaultSidebar = new ej.navigations.Sidebar({
  showBackdrop: true,
  type: "Push",
  width: '280px'
});
defaultSidebar.appendTo('#default-sidebar');
//end of sidebar initialization
// Toggle(Open/Close) the sidebar
document.getElementById('toggle').onclick = function() {
  defaultSidebar.toggle();
};
// Close the sidebar

```

```
document.getElementById('close').onclick = function() {
    defaultSidebar.hide();
};
//end of sidebar initialization
// Toggle(Open/Close) the sidebar
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- sidebar element declaration -->
    <aside id="default-sidebar">
      <div class="title"> Sidebar content</div>
      <div class="sub-title">
        Click the button to close the Sidebar.
      </div>
      <div class="center-align">
        <button id="close" class="e-btn close-btn">Close
Sidebar</button>
      </div>
    </aside>
    <!-- end of sidebar element -->
    <!-- main content declaration -->
    <div>
      <div class="title">Main content</div>
      <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
      <div style="padding:20px" class="center-align">
        <button id="toggle" class="e-btn e-info">Toggle
Sidebar</button>
```

```

        </div>
    </div>
    <!--end of main content -->
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Position

Positioning the Sidebar to the right or left of the main content can be achieved by using the `position` property. If the position is not set, the Sidebar will expand from the left to the body element. `enablePersistence` will persist the component's state between page reloads. `change` event will be triggered when the state(expand/collapse) of the component is changed.

INDEX.JS

```

ej.base.enableRipple(true);
//sidebar initialization
var defaultSidebar = new ej.navigations.Sidebar({
    showBackdrop: true,
    enablePersistence: true,
    type: "Push",
    width: '280px'
});
defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
//toggle button initialization
var togglebtn = new ej.buttons.Button({iconCss: 'e-icons burg-icon',
isToggle: true, content: 'Open'}, '#toggle');
//Click Event for toggle button to trigger show/hide methods of Sidebar
document.getElementById('toggle').onclick = function(){
    if (document.getElementById('toggle').classList.contains('e-active')) {
        togglebtn.content = 'Close';
        defaultSidebar.show();
    } else {
        togglebtn.content = 'Open';
        defaultSidebar.hide();
    }
};
// Close the sidebar
document.getElementById('close').onclick = function() {
    defaultSidebar.hide();
    document.getElementById('toggle').classList.remove('e-active');
    togglebtn.content = 'Open'
};
var positionLeft = new ej.buttons.RadioButton({ label: 'Left', name:
'state', checked: true, change: Change });
positionLeft.appendTo('#left');
//unchecked state

```



```

var positionRight = new ej.buttons.RadioButton({ label: 'Right', name:
'state', change: Change });
positionRight.appendTo('#right');
//change the position of the Sidebars
function Change(args){
defaultSidebar.position = (args.event.target.id == "left") ? "Left" :
"Right";
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- sidebar element declaration -->
    <aside id="default-sidebar">
      <div class="title"> Sidebar content</div>
      <div id="list"></div>
      <div class="sub-title">
        Click the button to close the Sidebar.
      </div>
      <div class="center-align">
        <button id="close" class="e-btn close-btn">Close
Sidebar</button>
      </div>
    </aside>
    <!-- end of sidebar element -->
    <!-- main content declaration -->
    <div id="head">
      <button id="toggle" class="e-btn e-info"></button>
    </div>
    <div class="maincontent" style="height:335px;border:1px solid gray">
      <div>
        <div class="title">Main content</div>

```

```

        <div class="sub-title">
            <div class="radiobutton">
                <input type="radio" id="left">
            </div>
            <div class="radiobutton">
                <input type="radio" id="right">
            </div>
        </div>
    </div>
</div>
<!-- end of main content declaration -->
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Animate

Animation transitions can be set while expanding or collapsing the Sidebar using the `animate` property. By default, `animate` property is set to true. `enableRTL` will display the sidebar in the right-to-left direction.

INDEX.JS

```

ej.base.enableRipple(true);

var defaultSidebar = new ej.navigations.Sidebar({
    type: "Push",
    width: '280px',
    animate: false,
    enableRtl: true
});
defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
// Toggle(Open/Close) the sidebar
document.getElementById('toggle').onclick = function() {
    defaultSidebar.toggle();
};
// Close the sidebar
document.getElementById('close').onclick = function() {
    defaultSidebar.hide();
};
//end of sidebar initialization
// Toggle(Open/Close) the sidebar

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Sidebar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <aside id="default-sidebar">
            <div class="title"> Sidebar content</div>
            <div class="sub-title">
                Click the button to close the Sidebar
            </div>
            <div class="center-align">
                <button id="close" class="e-btn close-btn">Close
Sidebar</button>
            </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div>
            <div class="title">Main content</div>
            <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
            <div style="padding:20px" class="center-align">
                <button id="toggle" class="e-btn e-info">Toggle Sidebar</button>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Close on document click

Sidebar can be closed on document click by setting `closeOnDocumentClick` to true. If this property is not set, the Sidebar will not close on document click since its default value is false. Sidebar can be kept opened during rendering using `isOpen` property.

INDEX.JS

```
ej.base.enableRipple(true);

var defaultSidebar = new ej.navigations.Sidebar({
  type: "Push",
  width: '280px',
  closeOnDocumentClick: true,
  isOpen: true
});
defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
// Toggle(Open/Close) the sidebar
document.getElementById('toggle').onclick = function() {
  defaultSidebar.show();
};
//end of sidebar initialization
// Toggle(Open/Close) the sidebar
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <aside id="default-sidebar">
      <div class="title"> Sidebar content</div>
    </aside>
    <!-- end of sidebar element -->
    <!-- main content declaration -->
    <div>
      <div class="title">Main content</div>
```

```

        <div class="sub-title"> Click the button to open the
Sidebar.</div>
        <div style="padding:20px" class="center-align">
            <button id="toggle" class="e-btn e-info">Open
Sidebar</button>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable gestures

Expand or collapse the Sidebar while swiping in touch devices using `enableGestures` property. By default, `enableGestures` is set to true.

INDEX.JS

```

ej.base.enableRipple(true);

var defaultSidebar = new ej.navigations.Sidebar({
    type: "Push",
    width: '280px',
    enableGestures:false
});
defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
// Toggle(Open/Close) the sidebar
document.getElementById('toggle').onclick = function() {
    defaultSidebar.toggle();
};
// Close the sidebar
document.getElementById('close').onclick = function() {
    defaultSidebar.hide();
};
//end of sidebar initialization
// Toggle(Open/Close) the sidebar

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Sidebar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <aside id="default-sidebar">
            <div class="title"> Sidebar content</div>
            <div class="sub-title">
                Click the button to close the Sidebar
            </div>
            <div class="center-align">
                <button id="close" class="e-btn close-btn">Close
Sidebar</button>
            </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div>
            <div class="title">Main content</div>
            <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
            <div style="padding:20px" class="center-align">
                <button id="toggle" class="e-btn e-info">Toggle
Sidebar</button>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Sidebar with navigation menu](#)
- [Sidebar responsive panel](#)
- [Sidebar with listView](#)
- [Usecase sample](#)

Custom context in EJ2 JavaScript Sidebar control

Sidebar has a flexible option to make it initialize, target to any HTML element alongside of the main content of a web page.

By default, Sidebar initializes target to the body element. Using the [target](#) property, set target element to initialize the Sidebar inside any HTML element apart from the body element.

If required, `zIndex` can be set when sidebar act as overlay type.

In the following sample, sidebar is rendered context to a div container element which has the CSS class `e-main-content`.

INDEX.TS

```
import { Sidebar } from '@syncfusion/ej2-navigations';
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let defaultSidebar: Sidebar = new Sidebar({
    width: "280px",
    type: "Push",
    target: '.maincontent'
});
defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
//toggle button initialization
let togglebtn: Button = new Button({iconCss: 'e-icons burg-icon', isToggle:
true, content:'Open'}, '#toggle');
// Close the Sidebar
document.getElementById('close').onclick = (): void => {
    defaultSidebar.hide();
    document.getElementById('toggle').classList.remove('e-active');
    togglebtn.content = 'Open'
};
//Click Event.
document.getElementById('toggle').onclick = (): void => {
    if (document.getElementById('toggle').classList.contains('e-
active')) {
        togglebtn.content = 'Close';
        defaultSidebar.show();
    } else {
        togglebtn.content = 'Open';
        defaultSidebar.hide();
    }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Sidebar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/css/bootstrap.min.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- sidebar element declaration -->
        <aside id="default-sidebar">
            <div class="title"> Sidebar content</div>
            <div id="list"></div>
            <div class="sub-title">
                Click the button to close the Sidebar.
            </div>
            <div class="center-align">
                <button id="close" class="e-btn close-btn">Close
Sidebar</button>
            </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div id="head">
            <button id="toggle" class="e-btn e-info"></button>
        </div>
        <div class="maincontent" style="height:325px;border:1px solid gray">
            <div>
                <div class="title"> Main content</div>
                <div class="sub-title"> content goes here.</div>
            </div>
        </div>
        <!--end of main content -->
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Variations in EJ2 JavaScript Sidebar control

The Sidebar component's expand behaviour can be modified based on the purpose of use.

Expanding types of Sidebar

The Sidebar can be set to initialize based on four different types that are consistent with the main component as explained below. When `dataBind` is invoked, this applies the pending property changes immediately to the component.

Item	Description
Over	Sidebar floats over the main content area.
Push	Sidebar pushes the main content area to appear side-by-side, and shrinks the main content within the screen width.
Slide	Sidebar translates the x and y positions of main content area based on the Sidebar width. The main content area will not be adjusted within the screen width.
Auto	Sidebar with Over type in mobile resolution, and Push type in other higher resolutions.

Auto is the default expand mode. Sidebar with type **Auto** will always expand on initial rendering, irrespective of [enableDock](#) and [isOpen](#) properties.

In the following sample, the types of Sidebar are demonstrated.

INDEX.TS

```
import { Sidebar } from '@syncfusion/ej2-navigations';
import { Button, RadioButton, ChangeArgs } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let defaultSidebar: Sidebar = new Sidebar({
    type: "Push",
    target: <HTMLElement>document.querySelector('.maincontent'),
});
defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
//toggle button initialization
let togglebtn: Button = new Button({iconCss: 'e-icons burg-icon', isToggle:
true, content:'Open'}, '#toggle');
//Click Event.
document.getElementById('toggle').onclick = (): void => {
    if (document.getElementById('toggle').classList.contains('e-
active')) {
        togglebtn.content = 'Close';
        defaultSidebar.show();
    } else {
        togglebtn.content = 'Open';
        defaultSidebar.hide();
    }
};
// Close the Sidebar
document.getElementById('close').onclick = (): void => {
    defaultSidebar.hide();
    document.getElementById('toggle').classList.remove('e-active');
    togglebtn.content = 'Open'
};
let typeOver: RadioButton = new RadioButton({ label: 'Over', name: 'state',
change: Change });
```

```

typeOver.appendTo('#over');
//unchecked state.
let typePush: RadioButton = new RadioButton({ label: 'Push', name: 'state',
checked: true, change: Change });
typePush.appendTo('#push');
let typeSlide: RadioButton = new RadioButton({ label: 'Slide', name:
'state', change: Change });
typeSlide.appendTo('#slide');
//unchecked state.
let typeAuto: RadioButton = new RadioButton({ label: 'Auto', name: 'state',
change: Change });
typeAuto.appendTo('#auto');
//change the togglebtn state.
function Changestate() {
    if (defaultSidebar.type == "Auto") {
        document.getElementById('toggle').classList.add('e-active');
        togglebtn.content = 'close';
    }
    else {
        document.getElementById('toggle').classList.remove('e-active');
        togglebtn.content = 'Open';
    }
}
function Change(args: ChangeArgs) {
    if ((<HTMLInputElement>args.event.target).id == "over") {
        defaultSidebar.type = "Over";
    } else if ((<HTMLInputElement>args.event.target).id == "push") {
        defaultSidebar.type = "Push";
    }
    else if ((<HTMLInputElement>args.event.target).id == "slide") {
        defaultSidebar.type = "Slide";
    }
    else {
        defaultSidebar.type = "Auto";
    }
    Changestate();
    defaultSidebar.dataBind();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Sidebar</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/css/bootstrap.min.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Sidebar element declaration-->
        <aside id="default-sidebar">
            <div class="title"> Sidebar content</div>
            <div id="list"></div>
            <div class="sub-title">
                Click the button to close the Sidebar.
            </div>
            <div class="center-align">
                <button id="close" class="e-btn close-btn">Close
Sidebar</button>
            </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div id="head">
            <button id="toggle" class="e-btn e-info"></button>
        </div>
        <div class="maincontent" style="height:335px;border:1px solid
gray">
            <div class="content">
                <div class="title">Main content</div>
                <div class="sub-title">
                    <div class="radiobutton">
                        <input type="radio" id="over">
                    </div>
                    <div class="radiobutton">
                        <input type="radio" id="push">
                    </div>
                    <div class="radiobutton">
                        <input type="radio" id="slide">
                    </div>
                    <div class="radiobutton">
                        <input type="radio" id="auto">
                    </div>
                </div>
            </div>
        </div>
        <!-- end of main content -->
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to add sidebar with custom animation](#)
- [How to add multiple sidebar](#)

Auto close in EJ2 JavaScript Sidebar control

The Sidebar often behaves differently on mobile display and differently on desktop display. It has an effective feature that offers to set it in opened or closed state depending on the specified resolution. This is achieved through [mediaQuery](#) property that allows you to set the Sidebar in an expanded state or collapsed state only in user-defined resolution.

In the following sample, `mediaQuery` has been used to close and open the Sidebar based on a specific resolution.

INDEX.TS

```

import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let defaultSidebar: Sidebar = new Sidebar({
  mediaQuery: window.matchMedia('(min-width: 600px)'), width: "280px",
});
defaultSidebar.appendTo('#default');
//end of Sidebar initialization
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Sidebar element declaration -->
        <aside id="default">
            <div class="title"> Sidebar </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div>
            <div class="title">Main content</div>
            <div class="sub-title">* Sidebar will collapse in mobile mode
automatically</div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

- In the following sample,the Sidebar will be in an expanded state only in resolution below 400px.

INDEX.TS

```

import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//start of sidebar initialization
let defaultSidebar: Sidebar = new Sidebar({
    mediaQuery: window.matchMedia('(max-width: 400px)'), width: "280px",
});
defaultSidebar.appendTo('#default');
//end of sidebar initialization
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Sidebar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/css/bootstrap.min.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- sidebar element -->
    <aside id="default">
      <div class="title"> Sidebar </div>
    </aside>
    <!-- end of sidebar element -->
    <!-- main content declaration -->
    <div>
      <div class="title">Main content</div>
      <div class="sub-title"> * Sidebar will open in mobile mode
automatically</div>
    </div>
    <!-- end of main content -->
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Docking sidebar in EJ2 JavaScript Sidebar control

[Dock](#) state of the Sidebar reserves some space on the page that always remains in a visible state when the Sidebar is collapsed. It is used to show the short term of a content like icons alone instead of lengthy text. To achieve this, set `enableDock` as true along with required `dockSize`.

In the following sample, the list item has icon with text representation. On dock state, only icons in list will be visible to represent the hint of the hidden text content.

INDEX.TS

```

import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(false);
//Sidebar initialization
let dockBar: Sidebar = new Sidebar({
  width: '220px',
  dockSize: '72px',

```

```

    enableDock: true
  });
dockBar.appendTo('#dockSidebar');
//end of Sidebar initialization
document.getElementById('toggle').onclick = (): void => {
  dockBar.toggle();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- sidebar element declaration -->
    <aside id="dockSidebar">
      <div class="dock">
        <ul>
          <li class="sidebar-item" id="toggle">
            <span class="e-icons expand"></span>
            <span class="e-text" title="menu">Menu</span>
          </li>
          <li class="sidebar-item">
            <span class="e-icons home"></span>
            <span class="e-text" title="home">Home</span>
          </li>
          <li class="sidebar-item">
            <span class="e-icons profile"></span>
            <span class="e-text" title="profile">Profile</span>
          </li>
          <li class="sidebar-item">
            <span class="e-icons info"></span>
            <span class="e-text" title="info">Info</span>
          </li>
        </ul>
      </div>
    </aside>
  </div>

```

```

        </li>
        <li class="sidebar-item">
            <span class="e-icons settings"></span>
            <span class="e-text"
title="settings">Settings</span>
        </li>
    </ul>
</div>
</aside>
<!-- end of sidebar element -->
<!-- main content declaration -->
<div id="main-content container-fluid col-md-12 ">
    <div class="title">Main content</div>
    <div class="sub-title"> Click the expand icon to open and
collapse icons to close the Sidebar.</div>
</div>
<!--end of main content -->
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to add sidebar navigation](#)

Style in EJ2 JavaScript Sidebar control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user's preference.

Customizing the sidebar

Use the below CSS to customize the sidebar root element.

```

.e-sidebar {
background: #898b2b
}

```

Customizing the sidebar based on the positions

Use the below CSS to customize the left positioned sidebar.

```

.e-sidebar.e-left {
border-right: 2px solid red;
}

```



```
}  
,
```

Use the below CSS to customize the right positioned sidebar.

```
,
```

```
.e-sidebar.e-right {  
border-left: 2px solid red;  
}
```

```
,
```

Customizing the sidebar based on the active state

Use the below CSS to customize the open state of the left positioned sidebar.

```
,
```

```
.e-sidebar.e-left.e-open {  
transition: transform 2.5s ease;  
}
```

```
,
```

Use the below CSS to customize the open state of the right positioned sidebar.

```
,
```

```
.e-sidebar.e-right.e-open {  
transition: transform 2.5s ease;  
}
```

```
,
```

Use the below CSS to customize the closed state of the left positioned sidebar.

```
,
```

```
.e-sidebar.e-left.e-transition.e-close {  
transition: transform 2.5s ease, visibility 1200ms;  
}
```

```
,
```

Use the below CSS to customize the closed state of the right positioned sidebar.

```
,
```

```
.e-sidebar.e-right.e-transition.e-close {  
transition: transform 2.5s ease, visibility 1200ms;  
}
```

```
,
```

Customizing the sidebar with dock state

When you enable the Dock support, the "e-dock" class will be added to the root element. Based on that class, you can also customize all the above stated customization. Use the following CSS to customize the sidebar element with a dock state.

```
、  
  
.e-sidebar.e-dock {  
background: #2d323e;  
}  
、
```

Customizing the different types of sidebar

Use the below CSS to customize the auto type sidebar.

```
、  
  
.e-sidebar.e-left.e-auto {  
background-color: pink;  
}  
、
```

Use the below CSS to customize the push type sidebar.

```
、  
  
.e-sidebar.e-left.e-push {  
background-color: beige;  
}  
、
```

Use the below CSS to customize the over type sidebar.

```
、  
  
.e-sidebar.e-left.e-over {  
background-color: aqua;  
}  
、
```

Use the below CSS to customize the slide type sidebar.

```
、  
  
.e-sidebar.e-left.e-slide {  
background-color: green;  
}  
、
```

Customizing the backdrop of the sidebar

Use the below CSS to customize the backdrop of the sidebar.

,

```
.e-sidebar-overlay {
background-color: aqua;
}
```

,

Customizing the sidebar in the RTL direction

When you enable the RTL (right to left direction) support, the "e-rtl" class will be added to the root element. Based on that class, you can also customize all the above stated customization. Use the following CSS to customize the sidebar element in the RTL (right to left direction) mode.

,

```
.e-sidebar.e-left.e-rtl {
background-color: antiquewhite;
}
```

,

Prevent the animation transition for the Sidebar component

Use the below CSS to prevent the animation transition for the Sidebar component.

,

```
.e-sidebar-context .e-content-animation {
transition: none !important;
transform: none !important;
}
```

,

Accessibility in EJ2 JavaScript Sidebar component

The Sidebar component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Sidebar component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Sidebar component followed the [WAI-ARIA](#) patterns to meet accessibility standards. By default, the Sidebar utilizes the [complementary](#) role, with the option to modify the ARIA role based on provided attributes to the root element, depending on the specific use case.

If there are multiple complementary landmark roles or aside elements in a document, it is important to provide a label for each landmark using the `aria-label` attribute. Alternatively, if the aside has a descriptive title, it can be referenced using the `aria-labelledby` attribute. This label will help assistive technology users quickly understand the purpose of each landmark.

For optimal accessibility, it is recommended to incorporate a trigger component that is navigable via a keyboard, such as a button. If this is not feasible, inclusion of the `tabIndex` attribute becomes necessary. Furthermore, explicit handling of the `aria-expanded` attribute is required for the trigger element.

Keyboard interaction

The Sidebar component does not have any inbuilt keyboard interaction support. However, you can utilize the keyboard interactions of focusable elements within the Sidebar component.

Ensuring accessibility

The Sidebar component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Sidebar component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Sidebar component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Sidebar with list view in EJ2 JavaScript Sidebar control

Any HTML element can be placed in the Sidebar content area. Sidebar supports all types of HTML structures like `TreeView`, `ListView`, etc.

In the following example, the Sidebar is rendered with `ListView` component in its content area.

INDEX.TS

```
import { Sidebar } from '@syncfusion/ej2-navigations';
import { ListView } from '@syncfusion/ej2-lists';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize ListView component
let data: { [key: string]: Object }[] = [
    { text: 'Home', id: 'list-02' },
    { text: 'Offers', id: 'list-03' },
    { text: 'Support', id: 'list-04' },
    { text: 'Logout', id: 'list-06' }
];
let listInstance: ListView = new ListView({
    //Set defined data to dataSource property
    dataSource: data
});
//Render initialized ListView component
listInstance.appendTo('#list');
let defaultSidebar: Sidebar = new Sidebar({
    type: "Over", width: '100%'
});
defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
// Toggle(Open/Close) the Sidebar
document.getElementById('toggle').onclick = (): void => {
    defaultSidebar.toggle();
};
// Close the Sidebar
document.getElementById('close').onclick = (): void => {
    defaultSidebar.hide();
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Sidebar</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- sidebar element declaration -->
        <aside id="default-sidebar">
            <div class="title1">Menu</div>
            <div class="closebtn">
                <button id="close" class="e-btn close-btn">
                    <span id="innerclose" class="e-icons close-icon"></span>
                </button>
            </div>
            <div id="listcontainer">
                <div id="list"></div>
            </div>
            <div class="sub-title">
                * ListView component is placed inside the sidebar content
area.
            </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div>
            <div class="title2">Main content</div>
            <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
            <div style="padding:20px" class="center-align">
                <button id="toggle" class="e-btn e-info">Toggle
Sidebar</button>
            </div>
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";

```

```

}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open and close the sidebar in EJ2 JavaScript Sidebar control

Opening and closing the Sidebar can be achieved with built-in public methods.

- [show\(\)](#): Method to open the Sidebar.
- [hide\(\)](#): Method to close the Sidebar.
- [toggle\(\)](#): Method to toggle between open and close states of the Sidebar.

In the following sample, toggle method has been used to show or hide the Sidebar on button click.

INDEX.TS

```

import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Sidebar initialization
let defaultSidebar: Sidebar = new Sidebar({
    showBackdrop: false,
    open: function(e)
    {
        console.log("Sidebar is opened");
    },
    close: function(e)
    {
        console.log("Sidebar is closed");
    }
});
defaultSidebar.appendTo('#default');
//end of Sidebar initialization
// Open the Sidebar
document.getElementById('open').onclick = (): void => {
    defaultSidebar.show();
};
// Toggle(Open/Close) the Sidebar
document.getElementById('toggle').onclick = (): void => {
    defaultSidebar.toggle();
};
// Close the Sidebar
document.getElementById('close').onclick = (): void => {
    defaultSidebar.hide();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Sidebar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/css/bootstrap.min.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- sidebar element declaration-->
        <aside id="default">
            <div class="title"> Sidebar content</div>
            <div class="sub-title">
                Click the button to close the Sidebar.
            </div>
            <div class="center-align">
                <button id="close" class="e-btn close-btn">Close
Sidebar</button>
            </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div>
            <div class="title">Main content</div>
            <div class="sub-title"> Click the button to Open the
Sidebar.</div>
            <div style="padding:20px" class="center-align">
                <button id="open" class="e-btn e-info">Open Sidebar</button>
            </div>
            <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
            <div style="padding:20px" class="center-align">
                <button id="toggle" class="e-btn e-info">Toggle
Sidebar</button>
            </div>
        </div>
        <!--end of main content declaration -->
    </div><script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Top and bottom sidebar in EJ2 JavaScript Sidebar control

You can initialize Sidebar at the left and right positions by using the [position](#) property. It will automatically adjust the width of the main content.

You can also initialize Sidebar at the top and bottom positions in application level. For initializing Sidebar, you need to manually adjust the height of the main content.

In the following sample, the [toggle](#) method has been used to show or hide the top and bottom sidebar on button click.

INDEX.TS

```
import { Sidebar } from '@syncfusion/ej2-navigations';
import { Button } from '@syncfusion/ej2-buttons';
// Top sidebar initialization
let topSidebar: Sidebar = new Sidebar({ type: "Push", open:
top_sidebar_open, close: top_sidebar_close });
topSidebar.appendTo('#top-sidebar');
// Bottom sidebar initialization
let bottomSidebar: Sidebar = new Sidebar({ type: "Push", open:
bottom_sidebar_open, close: bottom_sidebar_close });
bottomSidebar.appendTo('#bottom-sidebar');
// Top sidebar toggle button initialization
let topBtn: Button = new Button({ cssClass: 'e-info' });
topBtn.appendTo('#top-btn');
// Bottom sidebar toggle button initialization
let bottomBtn: Button = new Button({ cssClass: 'e-info' });
bottomBtn.appendTo('#bottom-btn');
topBtn.element.onclick = ()=>{
    topSidebar.toggle();
}
bottomBtn.element.onclick = ()=>{
    bottomSidebar.toggle();
}
function top_sidebar_open() {
    let element: Element = document.getElementsByClassName("e-content-
animation")[0];
    (<HTMLElement>element).style.height =
((<HTMLElement>element).offsetHeight - 75) + "px";
    element.classList.add("top_content_animation");
    // Remove the e-left class in sidebar
    topSidebar.element.classList.remove("e-left");
    // Add the custom class to sidebar
    topSidebar.element.classList.add("top_sidebar");
}
function top_sidebar_close() {
    let element: Element = document.getElementsByClassName("e-content-
animation")[0];
    (<HTMLElement>element).style.height =
((<HTMLElement>element).offsetHeight + 75) + "px";
    element.classList.remove("top_content_animation");
}
function bottom_sidebar_open() {
    let element: Element = document.getElementsByClassName("e-content-
animation")[0];
```

```

    (<HTMLElement>element).style.height =
    ((<HTMLElement>element).offsetHeight - 75) + "px";
    element.classList.add("bottom_animation_content");
    // Remove the e-left class in sidebar
    bottomSidebar.element.classList.remove("e-left");
    // Add the custom class to sidebar
    bottomSidebar.element.classList.add("bottom_sidebar");
  }
  function bottom_sidebar_close() {
    let element: Element = document.getElementsByClassName("e-content-
    animation")[0];
    (<HTMLElement>element).style.height =
    ((<HTMLElement>element).offsetHeight + 75) + "px";
    element.classList.remove("bottom_animation_content");
  }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
  beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <aside id="top-sidebar">
      <div class="title">
        <div style="display:inline-block"> Top Sidebar </div>
      </div>
      <div class="content">
        Place your top sidebar primary content here.
      </div>
    </aside>
    <aside id="bottom-sidebar">
      <div class="title">
        <div style="display:inline-block"> Bottom Sidebar </div>

```

```
</div>
<div class="content">
    Place your bottom sidebar primary content here.
</div>
</aside>
<div class="e-main-content">
    <div class="sub-content">
        <p>Place your main content here.....</p>
        <div id="button-align">
            <button id="top-btn" class="toggle">Toggle Top
Sidebar</button>
        </div>
        <div id="button-align">
            <button id="bottom-btn" class="toggle">Toggle Bottom
Sidebar</button>
        </div>
    </div>
</div>
</div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Multiple sidebar in EJ2 JavaScript Sidebar control

Two Sidebars can be initialized in a web page with same main content.

Sidebar can be initialized on right side or left side of the main content using `position` property.

The HTML element with class name `e-main-content` will be considered as the main content and both the Sidebars will behave as side content to this main content area of a web page.

INDEX.TS

```
import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let leftSidebar: Sidebar = new Sidebar({
    width: "200px",
    type: 'Push'
});
leftSidebar.appendTo('#default');
//end of leftSidebar initialization
let rightSidebar: Sidebar = new Sidebar({
    width: "200px",
    position: 'Right',
    type: 'Push'
});
rightSidebar.appendTo('#default1');
//end of rightSidebar initialization
// Toggle (Open/Close) the Sidebar1
document.getElementById('toggle-btn').onclick = function () {
```

```

    leftSidebar.toggle();
};
// Toggle(Open/Close) the Sidebar2
document.getElementById('toggle-btn2').onclick = function () {
    rightSidebar.toggle();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- left sidebar element -->
    <aside id="default">
      <div class="title"> Left Sidebar content</div>
    </aside>
    <!-- end of left sidebar element -->
    <!-- right sidebar element -->
    <aside id="default1">
      <div class="title">Right Sidebar content</div>
    </aside>
    <!-- end of right sidebar element -->
    <!-- main content declaration -->
    <div class="e-main-content" style="font-size:16px;padding:100px"
;="">
      <p>Place your main content here.....</p>
      <button id="toggle-btn" class="e-btn">Toggle Sidebar1</button>
      <br>
      <br>
      <button id="toggle-btn2" class="e-btn">Toggle Sidebar2</button>
    </div>
    <!-- end of main content -->
  </div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Multiple sidebar in same position in EJ2 JavaScript Sidebar control

You can initialize Sidebar at the left positions by using the [position](#) property. It will automatically adjust the width of the main content.

You can also initialize the another sidebar on the same position by adjusting the width of the first sidebar.

The following example demonstrate how to align the multiple sidebar on the same position.

INDEX.TS

```

import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Dock Sidebar initialization
let dockBar: Sidebar = new Sidebar({
    width: '220px',
    dockSize: '72px',
    enableDock: true,
    type: 'Push',
    target: 'e-main-content',
    zIndex: 3000
});
dockBar.appendTo('#dockSidebar');
//end of DockSidebar initialization
let defaultSidebar: Sidebar = new Sidebar({
    target: 'e-main-content',
    type: "Push",
});
defaultSidebar.appendTo('#default-sidebar');
//end of DefaultSidebar initialization
//switch the expand and collapse state
document.getElementById('buttonClick').onclick = function () {
    defaultSidebar.toggle();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Sidebar</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- first sidebar element -->
        <aside id="default-sidebar">
            <div class="sub-title">
                <h3> First Sidebar</h3>
                Place your primary content here.
            </div>
        </aside>
        <!-- dock sidebar element -->
        <aside id="dockSidebar">
            <div class="dock">
                <ul>
                    <li class="sidebar-item" id="toggle">
                        <span class="e-icons expand"></span>
                        <span class="e-text" title="menu">Menu</span>
                    </li>
                    <li class="sidebar-item">
                        <span class="e-icons home"></span>
                        <span class="e-text" title="home">Home</span>
                    </li>
                    <li class="sidebar-item">
                        <span class="e-icons profile"></span>
                        <span class="e-text" title="profile">Profile</span>
                    </li>
                    <li class="sidebar-item">
                        <span class="e-icons info"></span>
                        <span class="e-text" title="info">Info</span>
                    </li>
                    <li class="sidebar-item">
                        <span class="e-icons settings"></span>
                        <span class="e-text"
title="settings">Settings</span>
                    </li>
                </ul>
            </div>
        </aside>
        <!-- main content declaration -->
        <div class="e-main-content container-fluid col-md-12">
            <div class="title">Main content</div>

```

```

        <div class="sub-title">
            <p>Click the radio button to switch the sidebar position</p>
            <div class="center-align">
                <div class="column">
                    <button class="e-btn e-info"
id="buttonClick">Open/close hidden Sidebar</button>
                </div>
            </div>
        </div>
    </div>
    <!-- end of main content -->
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sidebar with treeview in EJ2 JavaScript Sidebar control

The following example demonstrates how to render TreeView component inside the Sidebar with dock state and how to achieve expand and collapse the functionalities simultaneously in the sidebar and Treeview.

On collapse, the LI elements of TreeView show icons only to represent the short sign of the hidden text content. On expand, hidden text content will be set to be visible.

INDEX.TS

```

import { Sidebar, TreeView } from '@syncfusion/ej2-navigations';
import { ListView } from '@syncfusion/ej2-lists';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sidebarMenu: Sidebar = new Sidebar({
    width: '290px',
    dockSize: "44px",
    enableDock: true,
    target: '.main-content',
    mediaQuery: '(min-width: 600px)',
    created: onCreate,
    close: onClose
});
sidebarMenu.appendTo('#sidebar-treeview');
let data: { [key: string]: Object }[] = [
    {
        nodeId: '01', nodeText: 'Installation', iconCss: 'icon-microchip
icon',
        nodeChild: [
            { nodeId: '05-01', nodeText: 'Calendar', iconCss: 'icon-
circle-thin icon' },
            { nodeId: '05-02', nodeText: 'DatePicker', iconCss: 'icon-
circle-thin icon' },
            { nodeId: '05-03', nodeText: 'DateTimePicker', iconCss:
'icon-circle-thin icon' },

```

```

        { nodeId: '05-04', nodeText: 'DateRangePicker', iconCss:
'icon-circle-thin icon' },
        { nodeId: '05-05', nodeText: 'TimePicker', iconCss: 'icon-
circle-thin icon' },
        { nodeId: '05-06', nodeText: 'SideBar', iconCss: 'icon-
circle-thin icon' }
    ]
},
{
    nodeId: '02', nodeText: 'Deployment', iconCss: 'icon-thumbs-up-
alt icon',
},
{
    nodeId: '03', nodeText: 'Quick Start', iconCss: 'icon-docs
icon',
},
{
    nodeId: '04', nodeText: 'Components', iconCss: 'icon-th icon',
    nodeChild: [
        { nodeId: '04-01', nodeText: 'Calendar', iconCss: 'icon-
circle-thin icon' },
        { nodeId: '04-02', nodeText: 'DatePicker', iconCss: 'icon-
circle-thin icon' },
        { nodeId: '04-03', nodeText: 'DateTimePicker', iconCss:
'icon-circle-thin icon' },
        { nodeId: '04-04', nodeText: 'DateRangePicker', iconCss:
'icon-circle-thin icon' },
        { nodeId: '04-05', nodeText: 'TimePicker', iconCss: 'icon-
circle-thin icon' },
        { nodeId: '04-06', nodeText: 'SideBar', iconCss: 'icon-
circle-thin icon' }
    ]
},
{
    nodeId: '05', nodeText: 'API Reference', iconCss: 'icon-code
icon',
    nodeChild: [
        { nodeId: '05-01', nodeText: 'Calendar', iconCss: 'icon-
circle-thin icon' },
        { nodeId: '05-02', nodeText: 'DatePicker', iconCss: 'icon-
circle-thin icon' },
        { nodeId: '05-03', nodeText: 'DateTimePicker', iconCss:
'icon-circle-thin icon' },
        { nodeId: '05-04', nodeText: 'DateRangePicker', iconCss:
'icon-circle-thin icon' },
        { nodeId: '05-05', nodeText: 'TimePicker', iconCss: 'icon-
circle-thin icon' },
        { nodeId: '05-06', nodeText: 'SideBar', iconCss: 'icon-
circle-thin icon' }
    ]
},
{
    nodeId: '06', nodeText: 'Browser Compatibility', iconCss: 'icon-
chrome icon'
},
{

```



```

        nodeId: '07', nodeText: 'Upgrade Packages', iconCss: 'icon-up-
hand icon'
    },
    {
        nodeId: '08', nodeText: 'Release Notes', iconCss: 'icon-
bookmark-empty icon'
    },
    {
        nodeId: '09', nodeText: 'FAQ', iconCss: 'icon-help-circled icon'
    },
    {
        nodeId: '10', nodeText: 'License', iconCss: 'icon-doc-text icon'
    }
];
// TreeView initialization
let mainTreeView: TreeView = new TreeView({
    fields: { dataSource: data, id: 'nodeId', text: 'nodeText', child:
'nodeChild' }, expandOn: 'Click',
});
mainTreeView.appendTo('#main-treeview');
document.getElementById('hamburger').onclick = (): void => {
    if (sidebarMenu.isOpen)
    {
        sidebarMenu.hide();
        mainTreeView.collapseAll();
    }
    else {
        sidebarMenu.show();
        mainTreeView.expandAll();
    }
};
function onCreate(): void {
    this.element.style.visibility = '';
}
function onClose(args: any) {
    mainTreeView.collapseAll();
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- sidebar element declaration -->
        <div class="main-header" id="header-section">
            <ul class="header-list">
                <li class="float-left header-style icon-menu"
id="hamburger"></li>
                <li class="float-left header-style nav-pane"><b>Navigation
Pane</b></li>
                <li class="header-style float-right support border-
left"><b>Support</b></li>
            </ul>
        </div>
        <aside id="sidebar-treeview" style="visibility: hidden">
            <div class="main-menu">
                <div>
                    <ul id="main-treeview">
                    </ul>
                </div>
            </div>
        </aside>
        <!-- end of sidebar element -->
        <!-- main content declaration -->
        <div class="main-content" id="main-text">
            <div class="sidebar-content">
                <h2 class="sidebar-heading"> Responsive Sidebar With
TreeView</h2>
                <p class="paragraph-content">This is a graphical aid for
visualising and categorising the site, in the
                    style of an expandable and collapsable treeview
component. It auto-expands to display the node(s),
                        if any, corresponding to the currently viewed title,
highlighting that node(s) and its ancestors.
                            Load-on-demand when expanding nodes is available where
supported (most graphical browsers), falling
                                back to a full-page reload. MediaWiki-supported caching,
aside from squid, has been considered so
                                    that unnecessary re-downloads of content are avoided
where possible. The complete
                                        expanded/collapsed state of the treeview persists across
page views in most situations.</p>
                    <div class="line"></div>
                    <h2 class="sidebar-heading">Lorem Ipsum Dolor</h2>
                    <p class="paragraph-content">Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do
                        eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis

```

```

        nostrud exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat. Duis aute irure
        dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur.</p>
        <div class="line"></div>
        <h2 class="sidebar-heading"> Lorem Ipsum Dolor</h2>
        <p class="paragraph-content">Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod
        tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud
        exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in
        reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint
        occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.</p>
        <div class="line"></div>
        <h2 class="sidebar-heading"> Lorem Ipsum Dolor</h2>
        <p class="paragraph-content">Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod
        tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, quis nostrud
        exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in
        reprehenderit in voluptate velit esse cillum dolore eu
fugiat nulla pariatur. Excepteur sint
        occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum.</p>
    </div>
</div>
    <!-- end of main scontent element -->
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Fixed sidebar in EJ2 JavaScript Sidebar control

The Sidebar does not require any specific style to make it as a fixed one. By default, the Sidebar position will be in fixed state. The following example demonstrates that the Sidebar is rendered with a fixed position. The position of the Sidebar will not change when scrolling the main content area.

INDEX.TS

```

import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Sidebar Initialization
let defaultSidebar: Sidebar = new Sidebar({
    width: '260px',
    created: onCreate
});

```

```

defaultSidebar.appendTo('#default-sidebar');
//end of Sidebar initialization
document.getElementById('hamburger').onclick = (): void => {
    defaultSidebar.toggle();
}
function onCreate(): void {
    this.element.style.visibility = '';
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- declaration of sidebar element -->
    <div id="wrapper">
      <!-- sidebar element declaration -->
      <aside id="default-sidebar" style="visibility: hidden">
        <div class="sidebar-header header-cover" style="background-color:#0378d5">
          <div class="image-container">
            <div class="sidebar-image">
            </div>
          </div>
          <div style="padding: 0 0 5px 0;">
            <a class="sidebar-brand" href="#settings-dropdown">
              john.doe@gmail.com
            </a>
            <span class="sf-icon-down icon"></span>
          </div>
        </div>
        <!-- Sidebar navigation -->

```

```

        <ul class="nav sidebar-nav">
            <li>
                <a href="#">
                    <i class="sf-icon-sidebar sf-icon-file"></i>
                    <span class="e-text"> Inbox</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <i class="sf-icon-sidebar sf-icon-starred"></i>
                    <span class="e-text"> Starred</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <i class="sf-icon-sidebar sf-icon-recent"></i>
                    <span class="e-text"> Snoozed</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <i class="sf-icon-sidebar sf-icon-
important"></i>
                    <span class="e-text"> Important</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <i class="sf-icon-sidebar sf-icon-offline"></i>
                    <span class="e-text"> Sent</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <i class="sf-icon-sidebar sf-icon-backup"></i>
                    <span class="e-text"> Draft</span>
                </a>
            </li>
        </ul>
    </aside>
    <!-- end of sidebar element -->
    <!-- main content declaration -->
    <div>
        <div class="content">
            <div id="left">
                <span id="hamburger" class="e-icons menu
default"></span>
            </div>
            <div id="center">
                <span>Inbox</span>
            </div>
            <div id="right">
                <span class="sf-icon-search"></span>
            </div>
        </div>
    </div>

```

```

        <div class="e-control e-listview e-list-template e-
touch">
            <ul class="e-list-parent e-ul ">
                <li class="e-list-group-item e-level-1"
role="group" data-uid="group-list-item-Today" aria-level="1">
                    <div class="e-text-content"
role="presentation"><span class="" style="width: 100%; margin-left: 2%;
margin-top: -2%;">Today</span></div>
                </li>
                <li class="e-list-item">
                    <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                        <span class="e-avatar e-icon sf-icon-
profile"></span>
                        <div>
                            <span class="e-list-item-header e-
name">Albert Lives</span>
                            </div>
                            <span class="received e-list-content e-
second-heading">Opening for Sales Manager</span>
                            <span class="e-list-item-header sf-icon-
star">
                                <span class="e-list-text">Hello Uta
Morgan,</span></span></div>
                        </li>
                        <li class="e-list-item">
                            <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                <span class="e-avatar e-icon sf-icon-
profile"></span>
                                <div>
                                    <span class="e-list-item-header e-
name">
                                        Ila Russo</span>
                                    </div>
                                    <span class="received e-list-content e-
second-heading">Business dinner invitation
                                    </span>
                                    <span class="e-list-item-header sf-icon-
star">
                                        <span class="e-list-text">Hello
Jelani Moreno,</span></span></div>
                            </li>
                            <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                                    <div>
                                        <span class="e-list-item-header e-
name">
                                            Garth Owen</span>
                                        </div>
                                        <span class="received e-list-content e-
second-heading">Application for Job Title</span>

```

```

                                <span class="e-list-item-header sf-icon-
star">
                                <span class="e-list-text">Hello Ila
Russo,</span></span>
                                </div>
                                </li>
                                <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                <span class="e-avatar e-icon sf-icon-
profile"></span>
                                <div>
                                <span class="e-list-item-header e-
name">Ursula Patterson</span>
                                </div>
                                <span class="received e-list-content e-
second-heading">Hello Kerry Best,</span>
                                <span class="e-list-item-header sf-icon-
star">
                                <span class="e-list-text">Programmer
Position Application</span></span>
                                </div>
                                </li>
                                <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                <span class="e-avatar e-icon sf-icon-
profile"></span>
                                <div>
                                <span class="e-list-item-header e-
name">
                                Nichole Rivas</span>
                                </div>
                                <span class="received e-list-content e-
second-heading">Annual Conference</span>
                                <span class="e-list-item-header sf-icon-
star">
                                <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                                </div>
                                </li>
                                <li class="e-list-item">
                                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                                <span class="e-avatar e-icon sf-icon-
profile"></span>
                                <div>
                                <span class="e-list-item-header e-
name">
                                Nichole Rivas</span>
                                </div>
                                <span class="received e-list-content e-
second-heading">Annual Conference</span>
                                <span class="e-list-item-header sf-icon-
star">
                                <span class="e-list-text">Hi Igor
Mccoy,</span></span>

```

```

        </div>
    </li>
    <li class="e-list-item">
        <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
            <span class="e-avatar e-icon sf-icon-
profile"></span>
            <div>
                <span class="e-list-item-header e-
name">
                    Nichole Rivas</span>
                </div>
                <span class="received e-list-content e-
second-heading">Annual Conference</span>
                <span class="e-list-item-header sf-icon-
star">
                    <span class="e-list-text">Hi Igor
Mccoy,</span></span>
            </div>
        </li>
        <li class="e-list-item">
            <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                <span class="e-avatar e-icon sf-icon-
profile"></span>
                <div>
                    <span class="e-list-item-header e-
name">
                        Nichole Rivas</span>
                    </div>
                    <span class="received e-list-content e-
second-heading">Annual Conference</span>
                    <span class="e-list-item-header sf-icon-
star">
                        <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                </div>
            </li>
            <li class="e-list-item">
                <div class="e-list-wrapper e-list-avatar e-
list-multi-line">
                    <span class="e-avatar e-icon sf-icon-
profile"></span>
                    <div>
                        <span class="e-list-item-header e-
name">
                            Nichole Rivas</span>
                        </div>
                        <span class="received e-list-content e-
second-heading">Annual Conference</span>
                        <span class="e-list-item-header sf-icon-
star">
                            <span class="e-list-text">Hi Igor
Mccoy,</span></span>
                    </div>
                </li>
                <li class="e-list-item">

```



```

<div class="e-list-wrapper e-list-avatar e-
list-multi-line">
    <span class="e-avatar e-icon sf-icon-
profile"></span>
    <div>
        <span class="e-list-item-header e-
name">Ursula Patterson</span>
    </div>
    <span class="received e-list-content e-
second-heading">Hello Kerry Best,</span>
    <span class="e-list-item-header sf-icon-
star">
        <span class="e-list-text">Programmer
Position Application</span></span>
    </div>
</li>
</ul>
</div>
</div>
<!--end of main content declaration -->
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sidebar with variation animation in EJ2 JavaScript Sidebar control

In the following example, the sidebar is rendered with custom animation effects. Click the buttons available in the main content area to check how the custom animations works with sidebar.

Sidebar will automatically adjust expanding animation to match any custom size specified in **CSS** styles.

INDEX.TS

```

import { Sidebar } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sidebarElement: Sidebar = new Sidebar({
    showBackdrop: true,width: '280px', created: oncreate
});
sidebarElement.appendTo('#sidebar-element');
document.getElementById('zoom').onclick = (): void => {
    sidebarElement.show();
    sidebarElement.element.classList.add("w3-animate-zoom");
}
// Opendoor Effect
document.getElementById('open_door').onclick = (): void => {
    sidebarElement.show();
    let sidebar: Element =document.getElementsByClassName("e-sidebar-
overlay")[0];

```

```

        sidebar.classList.add("move");
    }
    //Bottom to Top
    document.getElementById('bottom_top').onclick = (): void => {
        sidebarElement.show();
        sidebarElement.element.classList.add("w3-animate-bottom");
    }
    //Rotate Sidebar
    document.getElementById('rotate').onclick = (): void => {
        sidebarElement.show();
        sidebarElement.element.classList.add("rotate");
    }
    //Sidebar with 3d Animation
    document.getElementById('rotate_3d').onclick = (): void => {
        sidebarElement.show();
        sidebarElement.element.classList.add("rotate_3d");
    }
    //Reverse Slide Out
    document.getElementById('reverse').onclick = (): void => {
        sidebarElement.show();
        sidebarElement.element.classList.add("reverse_slide_out");
    }
    // Close the Sidebar
    document.getElementById('close_btn').onclick = (): void => {
        sidebarElement.element.classList.remove("sidebar");
        sidebarElement.element.classList.remove("rotate");
        sidebarElement.element.classList.remove("w3-animate-zoom");
        sidebarElement.element.classList.remove("w3-animate-bottom");
        sidebarElement.element.classList.remove("rotate_3d");
        sidebarElement.element.classList.remove("reverse_slide_out");
        sidebarElement.hide();
    };
    //Remove the Flickering Effect
    function oncreate(): void {
        this.element.style.visibility = '';
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Sidebar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.3/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div>
    <aside id="sidebar-element" class="sidebar" style="visibility:
hidden">
      <div class="title"> Sidebar content</div>
      <div class="sub-title">
        * Sidebar is rendered with animation effect
      </div>
      <div class="center-align">
        <button id="close_btn" class="e-btn close-btn">Close
Sidebar</button>
      </div>
    </aside>
    <!-- end of sidebar element -->
    <!-- main content declaration -->
    <div id="default">
      <div class="title">Sidebar Transitions</div>
      <div class="sub-title"> * Click the below button to render the
Sidebar with animation effect.</div>
      <div style="padding:20px" class="center-align">
        <button id="zoom" class="e-btn e-info">Zoom Sidebar</button>
        <button id="open_door" class="e-btn e-info">Open Door
</button>
        <button id="bottom_top" class="e-btn e-info">Bottom to
Top</button>
      </div>
      <div style="padding:20px" class="center-align">
        <button id="rotate" class="e-btn e-info"> Rotate</button>
        <button id="rotate_3d" class="e-btn e-info"> Rotate
3D</button>
        <button id="reverse" class="e-btn e-info"> Reverse Slide
Out</button>
      </div>
    </div>
  </div>
</body>
</html>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

Signature

Customization in EJ2 JavaScript Signature control

The Signature control draws stroke/path using moveTo() and lineTo() methods to connect one or more points while drawing in canvas. The stroke width can be modified by using its color and width. And the background can be modified by using its background color and background image.

Stroke Width

The variable stroke width is based on the values of [maxStrokeWidth](#), [minStrokeWidth](#) and [velocity](#) for smoother and realistic signature. The default value of minimum stroke width is set as 0.5, maximum stroke width is set as 2.5 and velocity is set as 0.7.

In the following example, minimum stroke width is set as 0.5, maximum stroke width is set as 3 and velocity is set as 0.7.

INDEX.TS

```
import { Signature } from '@syncfusion/ej2-inputs';
let signature: Signature = new Signature({maxStrokeWidth:3, minStrokeWidth:
0.5, velocity: 0.7});
signature.appendTo('#signature');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Signature</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Signature Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <h4>Sign here</h4>
    <div id="signature-control">
      <canvas id="signature"></canvas>
    </div>
  </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#signature {
    border: 1px solid lightgray;
    height: 100%;
    width: 100%;
}
#signature-control {
    height: 300px;
    width: 550px;
}

```

Stroke Color

Color of the stroke can be specified by using [strokeColor](#) property and it accepts hexadecimal code, RGB, and text. The default value of this property is "#000000".

INDEX.TS

```

import { Signature } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
let signature: Signature = new Signature({});
signature.appendTo('#signature');
let button: Button = new Button({cssClass: 'e-primary'}, '#set');
button.element.onclick = function (e) {
    let color: string = document.getElementById('text').value;
    signature.strokeColor = color;
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Signature</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript Signature Component">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="signature-base-control">
            <div id="input">
                <input type="text" id="text" placeholder="Enter the Stroke
Color Value">
                <button id="set">Set Stroke Color</button>
            </div>
            <div id="signature-control">
                <canvas id="signature" style="height: 100%; width:
100%;"></canvas>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
}

```

```

width: 30%;
}
#signature {
border: 1px solid lightgray;
height: 100%;
width: 100%;
}
#signature-control {
height: 300px;
width: 550px;
}

```

Background Color

Background color of a signature can be specified by using [backgroundColor](#) property and it accepts hexadecimal code, RGB, and text. The default value of this property is “#ffffff”.

INDEX.TS

```

import { Signature } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
let signature: Signature = new Signature({});
signature.appendTo('#signature');
let button: Button = new Button({cssClass: 'e-primary', '#set'});
button.element.onclick = function (e) {
    let bgColor: string = document.getElementById('text').value;
    signature.backgroundColor = bgColor;
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Signature</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Signature Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="signature-base-control">
            <div id="input">
                <input type="text" id="text" placeholder="Enter the
Background Color Value">
                <button id="set">Set Background Color</button>
            </div>
            <div id="signature-control">
                <canvas id="signature" style="height: 100%; width:
100%;"></canvas>
            </div>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#signature-control {
    height: 250px;
}
#signature-base-control {
    height: 300px;
    width: 550px;
}
#signature {
    border: 1px solid lightgray;
}
#input {
    margin-bottom: 30px;
}
#text {
    height: 30px;
    width: 300px;
}

```



```
}

```

Background Image

Background image of a signature can be specified by using [backgroundImage](#) property. The background image can be set by either hosting the image in our local IIS or online image.

INDEX.TS

```
import { Signature } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
let signature: Signature = new Signature({});
signature.appendTo('#signature');
let button: Button = new Button({cssClass: 'e-primary'}, '#set');
button.element.onclick = function (e) {
    let url: string = document.getElementById('text').value;
    signature.backgroundImage = url;
};

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Signature</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Signature Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="signature-base-control">
      <div id="input">
        <input type="text" id="text" placeholder="Enter the URL of
the background Image">
        <button id="set">Set Background Image</button>
      </div>

```

```
<div id="signature-control">
  <canvas id="signature" style="height: 100%; width:
100%;"></canvas>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
#signature-control {
  height: 250px;
}
#signature-base-control {
  height: 300px;
  width: 550px;
}
#signature {
  border: 1px solid lightgray;
}
#input {
  margin-bottom: 30px;
}
#text {
  height: 30px;
  width: 300px;
}
```

See Also

- [Save With Background](#)

Open save in EJ2 JavaScript Signature control

The Signature control supports to open the signature by using hosted/online URL or base64. And it also supports various save options like image, base64, and blob.

Open Signature

The signature control opens a pre-drawn signature as either base64 or hosted/ online URL using the [load](#) method. It supports the PNG, JPEG, and SVG image's base64.

INDEX.TS

```
import { Signature } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
let signature: Signature = new Signature({});
signature.appendTo('#signature');
let button: Button = new Button({cssClass: 'e-primary', '#open'});
button.element.onclick = function (e) {
    let sign: string = document.getElementById('text').value;
    signature.load(sign);
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Signature</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Signature Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="signature-base-control">
      <div id="input">
        <input type="text" id="text" placeholder="Enter the Base64
or URL of signature">
        <button id="open">Open</button>
```

```

        </div>
        <div id="signature-control">
            <canvas id="signature" style="height: 100%; width:
100%;"></canvas>
        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#signature-control {
    height: 250px;
}
#signature-base-control {
    height: 300px;
    width: 550px;
}
#signature {
    border: 1px solid lightgray;
}
#input {
    margin-bottom: 30px;
}
#text {
    height: 30px;
    width: 400px;
}

```

Save Signature

The Signature control saves the signature as base64, blob, and image like PNG, JPEG, and SVG.

Save as Base64

The `getSignature` method is used to get the signature as base64 with the PNG, JPEG, and SVG type. This can be loaded to signature using [load](#) method.

INDEX.TS

```
import { Signature } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
import { Dialog } from '@syncfusion/ej2-popups';
let signature: Signature = new Signature({});
signature.appendTo('#signature');
let dialogObj: Dialog = new Dialog({
    header: 'Base64 of the signature',
    height: 'auto',
    animationSettings: { effect: 'Zoom', duration: 400 },
    showCloseIcon: true,
    width: '80%',
    visible: false
});
dialogObj.appendTo('#defaultdialog');
let button: Button = new Button({cssClass: `e-primary`, content: 'Save As Base64'}, '#save');
document.getElementById('save').onclick = (): void => {
    dialogObj.content = signature.getSignature();
    dialogObj.show();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Signature</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript Signature Component">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
    type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
    ="text/javascript"></script>
</head>
<body>

    <div id="container">
```

```

        <h4>Sign here</h4>
        <div id="signature-control">
            <canvas id="signature"></canvas>
        </div>
        <button id="save" class="e-btn e-primary">Save as Base64</button>
        <div id="defaultdialog"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#signature {
    border: 1px solid lightgray;
    height: 100%;
    width: 100%;
}
#signature-control {
    height: 250px;
    width: 550px;
}
#save {
    margin-top: 30px;
}

```

Save as Blob

The [saveAsBlob](#) method is used to save the signature as Blob. It is defined as the chunk of binary data stored as a single entity in a database system.

Save as Image

The [save](#) method is used to save the signature as an image. And it accepts file name and file type as parameter. The file type parameter supports PNG, JPEG, and SVG and the default file type is PNG.

INDEX.TS

```
import { Signature, SignatureFileType } from '@syncfusion/ej2-inputs';
```

```
import { SplitButton, ItemModel, MenuEventArgs } from '@syncfusion/ej2-splitbuttons';
let signature: Signature = new Signature({});
signature.appendTo('#signature');
let items: ItemModel[] = [
    {
        text: 'Png'
    },
    {
        text: 'Jpeg'
    },
    {
        text: 'Svg'
    }
];
let splitBtn: SplitButton = new SplitButton({iconCss: 'e-sign-icons e-save',
items: items, content: 'Save', select: onSelect }, '#save');
function onSelect(args: MenuEventArgs): void {
    signature.save(args.item.text as SignatureFileType, 'Signature');
}
document.getElementById('save').onclick = function () {
    signature.save();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Signature</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Signature Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div>
      <span>Sign here</span>
```

```

        <button id="save">SAVE</button>
    </div>
    <div id="signature-control">
        <canvas id="signature"></canvas>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#signature {
    border: 1px solid lightgray;
    height: 100%;
    width: 100%;
}
#signature-control {
    height: 300px;
    width: 550px;
}
.e-split-btn-wrapper {
    float: right;
    margin-bottom: 5px;
    margin-right: 50px;
}
@font-face {
    font-family: 'font-icons';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6Wka0AAADQAAAAANmhoZWElUQQLAAAArAAAACRobXR4KAAAAAAAAAYAAAA
obG9jYQowB4oAAAH4AAAAFm1heHABIAGEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCIId8
AAA3IAAAAJwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAACgABAAAAQAAC7rwy18
PPPUACwQAAAAAAN3B8l4AAAAA3cHyXgAAAAAD9AP0AAAACAACAAAAAAAAAAEAAAAKAXgADAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wDnCGQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAIAAADAAAAFAADAAEAAAAUAAQAOGA

```



```

    font-weight: normal;
    font-style: normal;
}
.e-sign-icons {
    font-family: 'font-icons' !important;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-save::before {
    content: '\e701';
}

```

Save with Background

The [saveWithBackground](#) property is used to save the signature with its background and its default value is true. So, by default the signature is saved with its background.

In the following sample, the background color is set as 'rgb(103 58 183)' and save with background as true.

INDEX.TS

```

import { Signature, SignatureFileType } from '@syncfusion/ej2-inputs';
import { SplitButton, ItemModel, MenuEventArgs } from '@syncfusion/ej2-splitbuttons';
let signature: Signature = new Signature({saveWithBackground: true,
backgroundColor:"rgb(103 58 183)"});
signature.appendTo('#signature');
let items: ItemModel[] = [
    {
        text: 'Png'
    },
    {
        text: 'Jpeg'
    },
    {
        text: 'Svg'
    }
];
let splitBtn: SplitButton = new SplitButton({iconCss: 'e-sign-icons e-save',
items: items, content: 'Save', select: onSelect }, '#save');
function onSelect(args: MenuEventArgs): void {
    signature.save(args.item.text as SignatureFileType, 'Signature');
}
document.getElementById('save').onclick = function () {
    signature.save();
}

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Signature</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="TypeScript Signature Component">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div>
            <span>Sign here</span>
            <button id="save">Save</button>
        </div>
        <div id="signature-control">
            <canvas id="signature"></canvas>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;

```

```
position: absolute;
top: 45%;
width: 30%;
}
#signature {
border: 1px solid lightgray;
height: 100%;
width: 100%;
}
#signature-control {
height: 300px;
width: 550px;
}
.e-split-btn-wrapper {
float: right;
margin-bottom: 5px;
margin-right: 50px;
}
@font-face {
font-family: 'font-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6Wka0AAADQAAAAANmhoZWEIUQQLAAAArAAAAACRobXR4KAAAAAAYAAAA
obG9jYQowB4oAAAH4AAAAFmlheHABIAGEAAABCAAAACBuYw1lbLYTYgAAC3wAAAJJcG9zdIlCIId8
AAA3IAAAAJwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAACgABAAAAQAAC7rwy18
PPPUACwQAAAAAAN3B8l4AAAAA3cHyXgAAAAAD9AP0AAAACAACAAAAAAAAAAAEAAAAKAXgADAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCGQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAA
EAAAAAABAAAAQAAAAEAAAAABAAAAAQAIAAADAAAAFAADAAEAAAAUAAQOgA
AAAYABAAABAAALnAeCK//8AAOcA5wT//wAAAAAAAAQAGAAgAAAAABAAIAAwAEAAUABgAHAAGACQAAAA
AAAA6AFoAiACyAOgCKAPQBFYEtgAAAAQAAAAAA/QD8wADAAsAGQAJAAABESERARUzNTMVITUjESE
RMxUzESMRIREjESMRFSERIZUjNSEDHv3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFx/uIBHgI8j4/
X1/7iAR5I/04BZv6aA1r8pkcDWUhhAAAAAwAAAAAD7gP0AAMABwAPAAAlFSE1EzM1IwEhESMRIRE
jA0T9d1p8fP78A96r/XKl8WVlAgP//Bkd6P7OATIAAAMAAAAAA/QD9AACAYAGQAANYUnNxcBJzc
HFz8DNS8HDwIMASTqO+kB0+qpbulyBQQCagQFpggJCQoJCQkMOuo66QHS6alu6XIICQoJCgkIpgc
EAWEBawQAAAAABAAAAAAD9APqAAIABgAKAA8AACUHNyUBJwElByc3AQMLCQEBN8ctAj/+lAMBbAF
PeaF6/XNQAVsCjf78nyzh+v6ZowFkC3ihd/3r/qxJAoABCwAAAgAAAAAD8wPoAB4AIgAAEW8HFR8
KMz8DFSE1IQE3CQI9BgSJBwcEAWICAwQHBwkLqgkJCQoJCQlGAo39iP7IPwE6AfH+xwGwBg0ODg8
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkaAgAAAAAD8wPqAEkBGgAAAR8FDwvVHxM/CjU
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEvBTUvFg8TA4M
GBAMCAQEBAQQHHBAKQCcDAwECAQEDAwQFBgYHCACJCAkICQkJCAkIBwgHBgYFBAMDAGECBQUHCQk
KDAwNDQ4ODw4dHBoiJv4aJgQCBAYGCAkLDA8ICakJCgoLDAkKCQkJCAkIEA4ODQwLCQkHBgUEEW
CAgcGBQUdAwIBAgIDBAQEUBGgYHBwcHBgcGBgYFBQUEAwMCAgEBAQICBAUFBgHBQIDAgMDJgc
HBwcGBwYGCwsJBwv+oAMCAQEBAUHChEVGRwVFhYWFxYgHxwTEBAODQUGBAUDAVwHBgUCAQIDBAU
DpAMEBgGKCw0PCAKJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0
tGhMVFWwMDQ4OCwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBgJCwwOGfEVFRMSEhAQDg4NDAsKCgk
IDwsKCglwChgODg8ODw4NDAAoFBAMDawEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQPBQYGBg
ICAKHBwcGBgYFBQQEBAKAgEBAgIDBAQEUBGgYHBwcICAgHBwcGBRoAFBUXDA0NKACFBAQCAgE
BAGQE/6gBAQFBgYNDxASEh4gISEXFhYUEXIIYFBAIBwQCAgICBAFbCQsNBggHCACICAgEoxgdHh4
eHh0cGgWMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAAUAAAAA/QD5AA5AI4
AswDaAXCAAAEzHw8VDwcvBj0BLxUlHxMDDwUvFz8BHwk/BTUVDDclHwclLws1PwYfBicXDwQvCzU
/DTmfAycPDh8KDwQdAR8XOWE/CBMfAx0BHw07AT8NPQIVLCMPAQmxBwgODg0LCwoJCACGBQQDAG
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwcICQgKGxwcHh8V/sMGBg8SExUXFxkgIB8fHx4dHBs
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAWEBbiAaGxwdHR4eCAGIBwYGAIBAgMFBSA
fHh0bGhgfwAEXBi0dIh8aFg/+1yAZHQ0LCQgHBQDAGEDAwQJDhERExUXGBoc6QUJCQCgGfxMPDg0
LCgkHBgUDAGEDBAMFBQYHCAGJDxAREhAQIyS5Dw4NDAsJCACFAwMDAQEBAGMGCAoLDRcaExh0BAM
```

```

CAQICAwQJDA8RExUXFxaGhIkJCMhIR8ODg4MCwsQDwkHBgYE1wMDAgEBAgMEBAUFBgCHBwgICAk
ICQgHCACGBgYFBAMDAgICAgMFBQYGCAGKCgoMDBIJBwgKCwsNDQ4QDyMkJEMdHhwdHBwcGxoaGRg
XFxYWFbQCTAEDBACICgsMDQ0PDxAQEBftBAUDAwMCAQEBAQIDAwMFBDCsIASWFgoKCgcHBgUFBAM
CBQQGCQsIqQoKExQTFBITERUSEA8NCwkGBQIF/ncEAWQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFRi
QCgcGvxowFBUTFBISAwEBaWUGBgYGBgYFBAQTFBQTFBUUG5gjAxkRGBgYFhIVFhUbDQ0LDAoKCgg
HBgUFAwIEAgEBaWQGBwoLWwEFBwGH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU
HBwkJCgsICAgJCAkSExMTFBuUFR8gFRnJCAgICQkJCQkJChMUFBQTFBMSEhEQChMQDQwIBgIBAQI
EBgUFBgUGAXQTEYWNWgJCAGHBWYGBgUEAwMCAgICAwMEBQYGBgCHCAGJCG0RERAQEBApDw4ODQw
LCgkKEwWNDQWNDQ0NDQ0ZGBUIdg4MCwsKCAGHBQUEAwEBAGMADAAAAAAD8gP0AagADAAQABQAGAA
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRrWGFp8JHRCZiIiLOIhkDBAYICgoGBgc
FDAoKCAyFAgEDBAYICsGBgYGDaoKCAyFAgFxDxYGrBMPHEgWBgoBEPyuRAPk/BwBr96cVT+yGUs
DIhMWBxwcChYQLgcGBgYGCChYJBgsLCQgHBQEBAQEBAcHCgsFBWYGCwsJCAcFAQEBCQYICsGBj8
QHAYGHw8WI1H9BQL7/GMD6AAAAAQAAAAA/QDqAAGADYAPQBBAABNxmVITUBJRUFCTsBPwk9AS8
KDwoLEQMHAwERAYERIQJJg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY
GBWYFBgoIBWQCAQEg7YL1/vY9A+j8GAfBqf7tQpYBR3oHBgYMCgkHAwICAQECAgMHCQoMBgYHBWY
GDAoJBWMCQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAEgDeAAEAAAAAAAAAAQA
AAEAAAAAAAAEACgABAAEAAAAAAAAIABwALAAEAAAAAAAAAMACgASAAEAAAAAAAAQACgAcAAEAAAAAAAU
ACwAmAAEAAAAAAAYACgAxAAEAAAAAAALAA7AAEAAAAAAASAEgBnAAMAAQQAIAAAAgB5AAMAAQQA
JAAEAFAB7AAMAAQQAIAIADgCPAAMAAQQAAMAFACdAAMAAQQAIAQAFACxAAMAAQQAIAUAFgDFAAM
AAQQAIAAYAFADbAAMAAQQAIAoAWADvAAMAAQQAIAAsAJAFHIGZvbnQtaWNvbnNSZWdlbGZyZm9udC1
pY29uc2ZvbnQtaWNvbnNWZXJzaW9uIDEuMGZvbnQtaWNvbnNGb250IGdlbmVYXRlZCB1c2luZyB
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGYAbwBuAHQALQBpAGM
AbwBuAHMAUgBlAGcAdQBsaGEAcgBmAG8AbgB0AC0AaQBjAG8AbgBzAGYAbwBuAHQALQBpAGMAbwB
uAHMAVgBlAHIAcWBPAG8AbgAgADEALgAwAGYAbwBuAHQALQBpAGMAbwBuAHMARgBvAG4AdAAgAGc
AZQBwAGUAcgBhAHQAZQBkACAADQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcWBPAG8AbgAgAE0AZQB
0AHIAbwAgAFMADABLAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcWBPAG8AbgAuAGMAbwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBAEFAQYBBwEIAQkBCgELAAd
zYXZlXzAyB3NhdmUtMDEHhZWRpdF8wMwdlZGl0XzAxBNsZWfYDHBhaW50LWJlY2tldA9wYWLudC1
idWNrZXQtZGw1hZ2VzC3BpY3RlcmVzLXdmAAAA) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-sign-icons {
font-family: 'font-icons' !important;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-save::before {
content: '\e701';
}

```

Draw in EJ2 JavaScript Signature control

Draw

The [draw](#) method is used to draw a text as signature with different font families like Arial, Serif, with different font sizes. It accepts text, font family, font size as its parameters. The default font family is “Arial”, and the default font size is “30”.

INDEX.TS

```

import { Signature } from '@syncfusion/ej2-inputs';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Button } from '@syncfusion/ej2-buttons';
let signature: Signature = new Signature({});
signature.appendTo('#signature');
let ddlObj: DropDownList = new DropDownList(
    {
        value: 'Arial',
        popupHeight: '200px',
    });
ddlObj.appendTo('#ddl');
let ddlObj1: DropDownList = new DropDownList(
    {
        value: '20',
        popupHeight: '200px',
    });
ddlObj1.appendTo('#ddl1');
let button: Button = new Button({cssClass: 'e-primary'}, '#draw');
button.element.onclick = function (e) {
    let text: string = document.getElementById('text').value;
    let font: string = ddlObj.value;
    let size: number = ddlObj1.value;
    signature.draw(text, font, size);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Signature</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript Signature Component">
    <meta name="author" content="Syncfusion">
    <link href="styles.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="signature-control">
        <canvas id="signature" style="height: 100%; width:
100%;"></canvas>
      </div>
      <div id="input">
        <table>
          <tbody>
            <tr>
              <td><div>Enter the Text:</div></td>
              <td><input type="text" id="text" placeholder="Enter
the Text"></td>
            </tr>
            <tr>
              <td style="padding-top:10px"><div>Font
Family:</div></td>
              <td style="padding-top:10px">
                <div style="max-width: 200px">
                  <select id="ddl">
                    <option value="Arial">Arial</option>
                    <option value="Serif">Serif</option>
                    <option value="Sans-serif">Sans-
serif</option>
                    <option value="Cursive">Cursive</option>
                    <option value="Fantasy">Fantasy</option>
                  </select>
                </div></td>
            </tr>
            <tr>
              <td style="padding-top:10px"><div>Font
Size:</div></td>
              <td style="padding-top:10px">
                <div style="max-width: 200px">
                  <select id="ddl1">
                    <option value="20">20</option>
                    <option value="30">30</option>
                    <option value="40">40</option>
                    <option value="50">50</option>
                  </select>
                </div></td>
            </tr>
          </tbody>
        </table>
        <br>
        <button id="draw">Draw</button>
      </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
  </body></html>

```

STYLES.CSS

```
#container {
  visibility: hidden;
  text-align: center;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
#signature-control {
  height: 300px;
  width: 500px;
  float: left;
}
#signature {
  border: 1px solid lightgray;
}
#input {
  margin-top: 30px;
  margin-right: 20px;
  float: right;
}
#text {
  height: 30px;
}
```

User interaction in EJ2 JavaScript Signature control

The below interactions were available in Signature, and we can walk through one by one.

- Undo and Redo
- Clear
- Disabled
- ReadOnly

Undo and Redo

In the Signature, every action can be maintained as a snap for undo and redo operations. And maintained SnapIndex for indexing the snap collection.

The [undo](#) method reverts the last action of signature by decreasing SnapIndex value to index previous snap. Here, [canUndo](#) method is used to ensure whether undo can be performed or not.

The [redo](#) method reverts the last undo action of the signature by increasing the SnapIndex to get the next snap. Here, [canRedo](#) method is used to ensure whether redo can be performed or not.

Clear

The [clear](#) method is used to clear the signature and makes the canvas empty. This is also considered in Undo/ Redo. Here, [isEmpty](#) method is used to ensure whether the signature is empty or not.

Disabled

The [disabled](#) property is used to enable/disable the signature control. In the disabled state, the user is not allowed to draw signature. And it can't be focused until the user enables the signature.

ReadOnly

The [isReadOnly](#) property is used to enable/disable the ReadOnly Signature. It can be focused but it prevents drawing in Signature.

The following sample explains about user interactions available in signature.

INDEX.TS

```
import { Signature, SignatureChangeEventArgs } from '@syncfusion/ej2-inputs';
import { CheckBox, ChangeEventArgs, Button } from '@syncfusion/ej2-buttons';
let signature: Signature = new Signature({change: changeEvent});
signature.appendTo('#signature');
let undoButton: Button = new Button({cssClass: 'e-primary', disabled: true},
'#undo');
undoButton.element.onclick = function (e) {
    if (!signature.disabled && !signature.isReadOnly) {
        signature.undo();
    }
};
let redoButton: Button = new Button({cssClass: 'e-primary', disabled: true},
'#redo');
redoButton.element.onclick = function (e) {
    if (!signature.disabled && !signature.isReadOnly) {
        signature.redo();
    }
};
let clearButton: Button = new Button({cssClass: 'e-primary', disabled:
true}, '#clear');
clearButton.element.onclick = function (e) {
    if (!signature.disabled && !signature.isReadOnly) {
        signature.clear();
    }
};
let checkBox1: CheckBox = new CheckBox({ label: 'Disabled', change:
onChangeDisable}, '#disable');
function onChangeDisable(args: ChangeEventArgs): void {
    signature.disabled = args.checked;
}
let checkBox2: CheckBox = new CheckBox({ label: 'ReadOnly', change:
onChangeReadOnly}, '#readonly');
function onChangeReadOnly(args: ChangeEventArgs): void {
    signature.isReadOnly = args.checked;
}
function changeEvent(args: SignatureChangeEventArgs) {
    if (!signature.disabled && !signature.isReadOnly) {
        if (signature.canUndo()) {
            undoButton.disabled = false;
        } else {
```

```

        undoButton.disabled = true;
    }
    if (signature.canRedo()) {
        redoButton.disabled = false;
    } else {
        redoButton.disabled = true;
    }
    if (!signature.isEmpty()) {
        clearButton.disabled = false;
    } else {
        clearButton.disabled = true;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Signature</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Signature Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="option">
      <table>
        <tbody><tr>
          <td>
            <button id="undo" style="margin-right:
20px;">UNDO</button>
          </td><td>

```

```

                <button id="redo" style="margin-right:
20px;">REDO</button>
                </td><td>
                <button id="clear" style="margin-right:
200px;">CLEAR</button>
                </td><td>
                <div style="margin-bottom: 5px;"><input id="disable"
type="checkbox"></div>
                <div><input id="readonly" type="checkbox"></div>
                </td>
            </tr>
        </tbody></table>
    </div>
    <div id="signature-control">
        <canvas id="signature" style="height: 100%; width:
100%;"></canvas>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#signature-control {
    height: 300px;
    width: 550px;
}
#signature {
    border: 1px solid lightgray;
}
#option {
    margin-bottom: 10px;
}

```

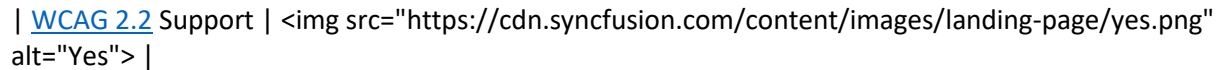
Accessibility in EJ2 JavaScript Signature control

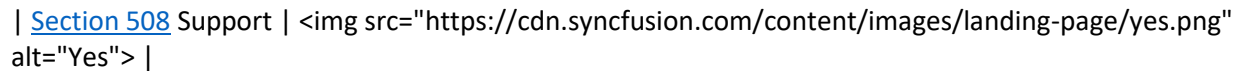
The Signature component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

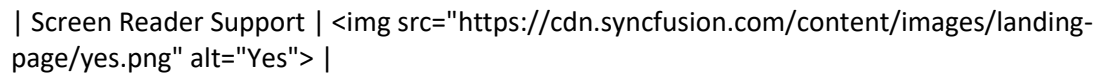
The accessibility compliance for the Signature component is outlined below.

| Accessibility Criteria | Compatibility |

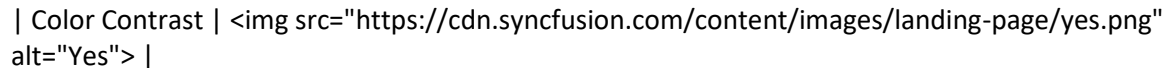
| -- | -- |

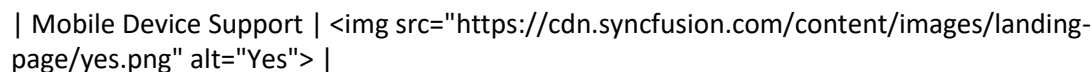
| [WCAG 2.2](#) Support |  |

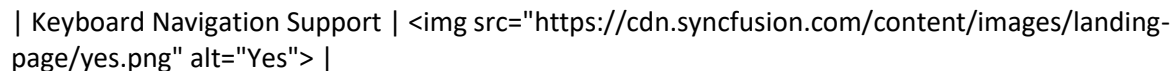
| [Section 508](#) Support |  |

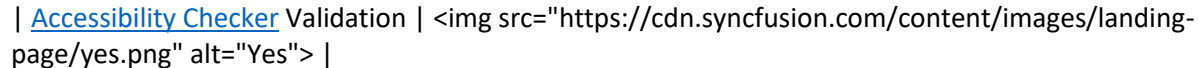
| Screen Reader Support |  |

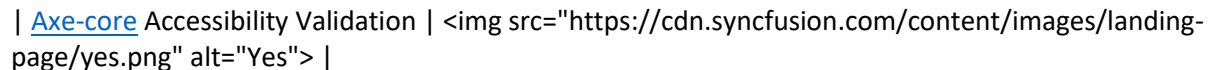
| Right-To-Left Support | Not Applicable |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

Keyboard interaction

The Signature component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Signature component.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + Z** | **Undo the last action.** |

| **Ctrl + Y** | **Redo the last action.** |

| **Ctrl + S** | **To save the signature.** |

| **delete** | **Erases all the signature strokes signed by user.** |

Ensuring accessibility

The Signature component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Signature component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Signature component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Toolbar integration in EJ2 JavaScript Signature control

The Signature control integrates with the toolbar and the interaction performed using the `change` event of the toolbar. In that, [canUndo](#), [canRedo](#) and [isEmpty](#) methods were used to enable/disable undo, redo, and clear buttons.

INDEX.TS

```
import { Toolbar, ClickEventArgs } from '@syncfusion/ej2-navigations';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
import { SplitButton, ItemModel, MenuEventArgs } from '@syncfusion/ej2-splitbuttons';
import { Button } from '@syncfusion/ej2-buttons';
import { Signature, ColorPicker, ColorPickerEventArgs, PaletteTileEventArgs, SignatureFileType } from '@syncfusion/ej2-inputs';
import { addClass, createElement, getComponent, enableRipple } from '@syncfusion/ej2-base';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
enableRipple(true);
let signature: Signature = new Signature({
  maxStrokeWidth: 2,
  change: function() {
    if (!signature.isEmpty()) {
      var saveBtn: SplitButton =
getComponent(document.getElementById("save-option"), 'split-btn');
      clearButton();
      saveBtn.disabled = false;
    }
  }
});
```

```

        }
        updateUndoRedo();
    }
});
signature.appendTo('#signature');
let items: ItemModel[] = [
    {
        text: 'Png'
    },
    {
        text: 'Jpeg'
    },
    {
        text: 'Svg'
    }
];
let toolbarObj: Toolbar = new Toolbar({
    width: '100%',
    items: [
        { text: 'Undo', prefixIcon: 'e-icons e-undo', tooltipText: 'Undo (Ctrl + Z)' },
        { text: 'Redo', prefixIcon: 'e-icons e-redo', tooltipText: 'Redo (Ctrl + Y)' },
        { type: 'Separator' },
        { tooltipText: 'Save (Ctrl + S)', type: 'Button', template: '<button id="save-option"></button>' },
        { type: 'Separator' },
        { tooltipText: 'Stroke Color', type: 'Input', template: '<input id="stroke-color" type="color"/>' },
        { type: 'Separator' },
        { tooltipText: 'Background Color', type: 'Input', template: '<input id="bg-color" type="color"/>' },
        { type: 'Separator' },
        { tooltipText: 'Stroke Width', type: 'Input', template: '<input id="stroke-width" type="text"/>' },
        { type: 'Separator' },
        { text: 'Clear', prefixIcon: 'e-sign-icons e-clear', tooltipText: 'Clear' },
        { tooltipText: 'Disabled', type: 'Input', template: new CheckBox({ label: 'Disabled', checked: false, change: disabledChange}), align: 'Right' }
    ],
    created: () => {
        let ddl: DropDownList = new DropDownList({
            dataSource: [1, 2, 3, 4, 5],
            width: '60',
            value: 2,
            change: function(args) {
                signature.maxStrokeWidth = args.value;
            }
        });
        ddl.appendTo('#stroke-width');
        new SplitButton({ iconCss: 'e-sign-icons e-save', content: 'Save',
items: items, select: onSelect, disabled: true }, '#save-option');
        let strokeColor: ColorPicker = new ColorPicker({
            modeSwitcher: false,
            columns: 4,
            presetColors: {

```

```

        'custom': ['#000000', '#000000', '#e91e63', '#9c27b0',
        '#673ab7', '#2196f3', '#03a9f4', '#00bcd4',
        '#009688', '#8bc34a', '#cddc39', '#ffeb3b']
    },
    beforeTileRender: (args: PaletteTileEventArgs) => {
        args.element.classList.add('e-circle-palette');
        args.element.appendChild(createElement('span', { className:
'e-circle-selection' }));
    },
    showButtons: false, mode: 'Palette', cssClass: 'e-stroke-color',
change: strokeColorChanged});
strokeColor.appendTo('#stroke-color');
let bgColor: ColorPicker = new ColorPicker({
    modeSwitcher: false,
    columns: 4,
    presetColors: {
        'custom': ['#ffffff', '#f44336', '#e91e63', '#9c27b0',
        '#673ab7', '#2196f3', '#03a9f4', '#00bcd4',
        '#009688', '#8bc34a', '#cddc39', '#ffeb3b']
    },
    beforeTileRender: (args: PaletteTileEventArgs) => {
        args.element.classList.add('e-circle-palette');
        args.element.appendChild(createElement('span', { className:
'e-circle-selection' }));
    },
    showButtons: false, mode: 'Palette', cssClass: 'e-bg-color',
noColor: true, change: bgColorChanged});
bgColor.appendTo('#bg-color');
addClass([strokeColor.element.nextElementSibling.querySelector('.e-
selected-color')], 'e-sign-icons');
addClass([bgColor.element.nextElementSibling.querySelector('.e-
selected-color')], 'e-sign-icons');
clearButton();
let toolbarItems: NodeList<Element> =
document.querySelectorAll('.e-toolbar .e-toolbar-items .e-toolbar-item .e-
tbar-btn.e-tbtn-txt');
for (var i = 0; i < toolbarItems.length; i++) {
    if (toolbarItems[i].children[0].classList.contains('e-undo')) {
        let undoButton: Button = getComponent(toolbarItems[i] as
HTMLElement, 'btn');
        undoButton.disabled = true;
    }
    if (toolbarItems[i].children[0].classList.contains('e-redo')) {
        let redoButton: Button = getComponent(toolbarItems[i] as
HTMLElement, 'btn');
        redoButton.disabled = true;
    }
}
},
clicked: (args: ClickEventArgs) => {
    let saveBtn: SplitButton =
getComponent(document.getElementById("save-option"), 'split-btn');
    if (signature.disabled && args.item.tooltipText != 'Disabled') {
        return;
    }
    switch (args.item.tooltipText) {
        case 'Undo (Ctrl + Z)':

```

```

        if (signature.canUndo()) {
            signature.undo();
            updateUndoRedo();
            updateSaveBtn();
        }
        break;
    case 'Redo (Ctrl + Y)':
        if (signature.canRedo()) {
            signature.redo();
            updateUndoRedo();
            updateSaveBtn();
        }
        break;
    case 'Clear':
        signature.clear();
        if (signature.isEmpty()) {
            clearButton();
            saveBtn.disabled = true;
        }
        break;
    }
}
});
toolbarObj.appendTo('#toolbar');
function disabledChange(args: ChangeEventArgs): void {
    signature.disabled = args.checked;
}
function updateSaveBtn() {
    let saveBtn: SplitButton = getComponent(document.getElementById("save-
option"), 'split-btn');
    if (signature.isEmpty()) {
        saveBtn.disabled = true;
    }
}
function onSelect(args: MenuEventArgs): void {
    signature.save(args.item.text as SignatureFileType, 'Signature');
}
document.getElementById('save-option').onclick = function () {
    signature.save();
};
function strokeColorChanged(args: ColorPickerEventArgs): void {
    if (signature.disabled) {
        return;
    }
    let selElem: HTMLElement =
this.element.nextElementSibling.querySelector('.e-selected-color') as
HTMLElement;
    selElem.style.borderBottomColor = args.currentValue.rgba;
    signature.strokeColor = args.currentValue.rgba;
}
function bgColorChanged(args: ColorPickerEventArgs): void {
    if (signature.disabled) {
        return;
    }
    let selElem: HTMLElement =
this.element.nextElementSibling.querySelector('.e-selected-color') as
HTMLElement;

```



```

signature.backgroundColor = args.currentValue.rgba;
selElem.style.borderBottomColor = args.currentValue.rgba;
}
function clearButton() {
    let tlItems: NodeListOf<Element> = document.querySelectorAll('.e-toolbar
.e-toolbar-items .e-toolbar-item .e-tbar-btn.e-tbtn-txt');
    for (var i = 0; i < tlItems.length; i++) {
        if (tlItems[i].children[0].classList.contains('e-clear')) {
            let clrBtn: Button = getComponent(tlItems[i] as HTMLElement,
'btn');
            if (signature.isEmpty()) {
                clrBtn.disabled = true;
            } else {
                clrBtn.disabled = false;
            }
        }
    }
}
function updateUndoRedo() {
    let undoButton: Button; let redoButton: Button
    let tlItems: NodeListOf<Element> = document.querySelectorAll('.e-toolbar
.e-toolbar-items .e-toolbar-item .e-tbar-btn.e-tbtn-txt');
    for (var i = 0; i < tlItems.length; i++) {
        if (tlItems[i].children[0].classList.contains('e-undo')) {
            undoButton = getComponent(tlItems[i] as HTMLElement, 'btn');
        }
        if (tlItems[i].children[0].classList.contains('e-redo')) {
            redoButton = getComponent(tlItems[i] as HTMLElement, 'btn');
        }
    }
    if (signature.canUndo()) {
        undoButton.disabled = false;
    } else {
        undoButton.disabled = true;
    }
    if (signature.canRedo()) {
        redoButton.disabled = false;
    } else {
        redoButton.disabled = true;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Signature</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript Signature Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="signature-toolbar-control">
            <div id="toolbar" style="width: 100%;"></div>
            <div id="signature-control">
                <canvas id="signature" style="height: 100%; width:
100%;"></canvas>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
    font-family: 'font-icons';
    src:

```

```
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKIAAAAwAgTlMvMjltSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAIsaGVhZB6Wka0AAADQAAAAANmhoZWEIUQQLAAAARAAAACRobXR4KAAAAAAYAAAAA
obG9jYQowB4oAAAH4AAAAFmlheHABIAGEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCid8
AAA3IAAAAJwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAACGABAAAAQAAC7rwy18
PPPUACwQAAAAAAN3B8l4AAAAA3cHyXgAAAAAD9AP0AAAACAACAAAAAAAAAAAEAAAAKAXgADAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAaZABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCGQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAIAAADAAAAFAADAAEAAAAUAAQAOgA
AAAYABAABALnAecK//8AAOcA5wT//wAAAAAAQAGAAgAAAABAAIAAwAEAAUABgAHAAGACQAAAAA
AAAA6AFoAiACyAOgCKAPQBFYEtgAAAAQAAAAA/QD8wADAAsAGQajAAABESERARUzNTMVITUjESE
RMxUjESMRIREjESMRFSERIZUjNSEDHv3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFx/uIBHgI8j4/
X1/7iAR5I/O4BZv6aA1r8pkcDWUhhAAAAAwAAAAAD7gP0AAMABwAPAAAlFSE1EzM1IwEhESMRIRE
jA0T9d1p8fP78A96r/XK18WVlAgP//Bkd6P7OATIAAAAAAAAA/QD9AACAAYAGQAANYUnNxcBJzc
HFz8DNS8HDwIMASTq0+kB0+qpbulyBQQCAGQFPggJCQoJCQkM0uo66QHS6alu6XIICQoJCgkIpgc
EAWEBawQAAAAABAAAAAAD9APqAAIABgAKAA8AACUHNyUBJwElByc3AQm1CQEBN8ctAj/+laMBbAF
PeaF6/XNQAVsCjF78nyzH+v6ZowFkC3ihd/3r/qxJAoABCwAAAgAAAAAD8wPoAB4AIgAAEW8HFR8
KMz8DFSE1IQE3CQI9BgSJBwcEAWICAwQHBwkLqgkJCQoJCQlGAo39iP7IPwE6AfH+xwGwBg0ODg8
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkAAgAAAAAD8wPqAEkBGgAAAR8FDwVHxM/CjU
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEvBTUvFg8TA4M
GBAMCAQEBAQQHHBAKQCcDAwECAQEDAwQFBgYHCACJCAkICQkICAKIBwgHBgYFBAMDAGECBQUHCQk
KDAwNDQ4ODw4dHB0iJv4aJgQCBAYGCAkLDA8ICAKJCgoLDAkKCQkICAKIEA4ODQwLCQkHBgUEEWm
CAgcGBQUDAwIBAgIDBAQEBQUGBgYHBwCHBgcGBgYFBQUEAwMCAgEBAQICBAUFBgCHBQIDAgMDJgc
HBwCHBwYGCwsJBwv+oAMCAQEBAUHChEVGRwVfHxYWFxYgHxwTEBAODQUGBAUDAVwHBgUCAQIDBAU
DpAMEBggKCw0PCAKJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0
tGhMVfwwMDQ4OCwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBggJCwwOGfEVFRMSEhAQDg4NDAsKCgk
IDwsKCglwChgODg8ODw4NDaoFBAMDawEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQPBQYGBgg
ICAKHBwCHBwYFBQQEBAWCAgEBAgIDBAQEBQUGBgYHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBw
ICAgHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBw
BAGQE/6gBAQFBgYNDxASEh4gISEXFhYUEXiYFBAIBwQCAgICBAFbCQsNBggHCACICAgEoxgdHh4
eHh0cGgXMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAUAAAAA/QD5AA5AI4
AswDaAXcAAAEzHw8VDwcvBj0BLxULHxMDDwUvFz8BHwk/BTUVdDclHwclLws1PwYfBicXDwQvCzU
/DTmfAycPdH8KDwQdAR8XOWE/CBMfAx0BHW07AT8NPQIvLCMPAQmxBwgODg0LCwoJCACGBQQDAgE
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBwCHBw
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAWEBbiAaGxwDHR4eCAGIBwYGAIBAgMFBsA
fHh0bGhgFWAEXBi0dIh8aFg/+1yAZHQ0LCQgHBQQDAgEDAwQJDhERExUXGBoc6QUJCQCgGfxMPDg0
LCgkHBgUDAgEDBAMFBQYHCAGJDxAREhAQIyS5Dw4NDAsJCACFAwMDAQEBAGMGCAoLDRcaExh0BAM
CAQICAwQJDA8RExUXFxoAGhIkJCMhIR8ODg4MCwsQDwkHBgYE1wMDAgEBAgMEBAUFBgCHBwGICAK
ICQgHCACGBgYFBAMDAGICAgMFBQYGCAGKCgoMDBIJBwgKCwsNDQ4QDyMkJEMdHhwdHBwCHBwCHBw
XfYwYFBQCTAEDBACICgsMDQ0PDxQAEbftBAUDAwMCAQEBAQIDAwMFBDCsIASwFgoKCgCHBwUFBAM
CBQQGCQsIqQoKExQTFBITERUSEA8NCwkGBQIF/ncEAWQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFRi
QCgcGvxowFBUTFBISAwEBawUGBwYGBgYFBwYFBwYFBwYFBwYFBwYFBwYFBwYFBwYFBwYFBwYFBw
HBgUFAwIEAgEBAwQGBwOLWwEFBwGH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU
HBwkJCgsICAgJCAkSExMTFBUFR8gFRnJCAgICQkJCQkJCChMUFBQTFBMSHhEQChMQDQwIBgIBAQI
EBgUFBgUGAXQTEYwNwGJCAGHBwYGBgUEAwMCAgICAwMEBQYGBgCHCAGJCG0RERAQEBApDw4ODQw
LCgkKEwWNDQwNDQ0NDQ0ZGBUiDg4MCwsKCAGHBQUEAwEBAGMADAAAAAAD8gP0AAGADAAQABQAGAA
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRrWGFp8JHRCZiIiLOIhKDBAYICgoGBgc
FDIAOKCAYFAgEDBAYICQsGBgYGDaoKCAyFAgFxDxYGRBMPHEgWBgoBEPYURAPk/BwBr96cVT+yGUs
DIhMWBxwChYlGcGBgYGCYJBgsLCQgHBQEBQECBACyCgSFBwYGCwsJCACFAQECCBYICQsGBj8
QHAYGHw8WI1H9BQL7/GMD6AAAAAQAAAAA/QDQAAAGADYAPQBBAABNxmVITUBJRUFCTsBPwk9AS8
KDwOLeQMHAwERAYERIQJjg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY
GBwYFBgoIBwQCAQEg7YL1/vY9A+j8GAfBqf7tQpYBR3oHBgYMCgkHAWICAQECAgMHCQoMBgYHBwY
GDAoJBwMCAQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAAEACgABAAEAAAAAAAAIABwALAAEAAAAAAAAAMACgASAAEAAAAAAAAAQcAgAAEAAAAAAAU
ACwAmAAEAAAAAAAYACgAXAAEAAAAAAALAA7AAEAAAAAAASAEgBnAAMAAQQAIAAAAgB5AAMAAQQA
JAAEFAB7AAMAAQQAIAIADgCPAAMAAQQAIAAMAFACdAAMAAQQAIAAQAFACxAMAAQQAIAUAUFgDFAAM
AAQQAIAAYAFADbAAMAAQQAIAAoAWADvAAMAAQQAIAAsAJAFHIGZvbnQtaWNvbnNSZWdlbGZyZm9udC1
```

```

pY29uc2ZvbnQtaWNvbnNWZXJzaW9uIDEuMGZvbnQtaWNvbnNGb250IGd1bmVyYXRlZCB1c2luZyB
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgaGYAbwBuAHQALQBpAGM
AbwBuAHMAUgBlAGcAdQBsAGEAcgBmAG8AbgB0AC0AaQBJAG8AbgBzAGYAbwBuAHQALQBpAGMABwB
uAHMAVgBlAHIAcwBpAG8AbgAgADEALgAwAGYAbwBuAHQALQBpAGMABwBuAHMARgBvAG4AdAAgAGc
AZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdABlAGQAaQBVAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBBAEFAQYBBwEIAQkBCgELAAAd
zYXZlXzAyB3NhdmUtMDEHZWRpdF8wMwdlZG10XzAxBNsZWZyDHBhaW50LWJlY2tldA9wYWludC1
idWNrZXQtd2YgaW1hZ2VzC3BpY3RlcmVzLXdmAAAA) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-sign-icons {
    font-family: 'font-icons' !important;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
#toolbar-control {
    border: 1px solid lightgray;
}
#toolbar {
    border: none;
    border-bottom: 1px solid lightgray;
    box-sizing: border-box;
}
#toolbar-control .e-btn:disabled {
    opacity: 0.5 !important;
    pointer-events: none;
}
#signature-control {
    height: 300px;
    width: 100%;
    margin: 0;
}
#signature {
    border: 1px solid lightgray;
}
.e-colorpicker-wrapper.e-bg-color #bg-color+.e-split-btn-wrapper .e-split-
btn .e-selected-color {
    background: none;
    border-bottom-style: solid;
    border-bottom-width: 3px;
    width: 14px;
    margin: 0px 2px;
    border-bottom-color: #ffffff;
}
.e-colorpicker-wrapper.e-stroke-color #stroke-color+.e-split-btn-wrapper .e-
split-btn .e-selected-color {
    background: none;
    border-bottom-style: solid;
    border-bottom-width: 3px;
}

```

```

width: 14px;
margin: 0px 2px;
border-bottom-color: #000000;
}
.e-colorpicker-wrapper.e-bg-color #bg-color+.e-split-btn-wrapper .e-split-
btn .e-selected-color .e-split-preview,
.e-colorpicker-wrapper.e-stroke-color #stroke-color+.e-split-btn-wrapper .e-
split-btn .e-selected-color .e-split-preview {
  display: none;
}
.e-colorpicker-wrapper.e-bg-color #bg-color+.e-split-btn-wrapper .e-split-
btn .e-selected-color::before {
  content: '\e707';
}
.e-colorpicker-wrapper.e-stroke-color #stroke-color+.e-split-btn-wrapper .e-
split-btn .e-selected-color::before {
  content: '\e704';
}
.e-clear::before {
  content: '\e706';
}
.e-save::before {
  content: '\e701';
}
}
.e-container .e-palette .e-circle-palette {
  border: 0;
  height: 32px;
  width: 32px;
  border-radius: 20px;
  margin: 4px;
}
.e-container .e-palette .e-circle-palette:hover {
  box-shadow: none;
  transform: scale(1.2);
  transition: transform .2s ease-out;
}
.e-circle-palette .e-circle-selection {
  height: 32px;
  width: 32px;
  border-radius: 20px;
  display: inline-block;
  transform: scale(0);
  transition: transform 1.2s ease-in;
}
.e-circle-palette.e-selected .e-circle-selection {
  transform: scale(0.8);
  background-color: #fff;
  transition: transform .2s ease-out;
}
}
#circle-palette+.e-container,
#scroll-palette+.e-container {
  background-color: transparent;
  border-color: transparent;
  box-shadow: none;
}
}

```

Skeleton

Shapes in EJ2 JavaScript Skeleton control

The Skeleton control support various built-in shape variants to design layout of the page. You can use the [shape](#) property to create a preview of any layout.

The Skeleton component supports the following shapes,

Circle skeleton shape

```
`js
// Initialize Skeleton control for circle shape
var circle = new ej.notifications.Skeleton({
  shape: 'Circle',
  width: "48px"
});
// Render initialized Skeleton.
circle.appendTo('#circleSkeleton');
`
```

Square skeleton shape

```
`js
// Initialize Skeleton control for square shape
var square = new ej.notifications.Skeleton({
  shape: 'Square',
  width: "48px"
});
// Render initialized Skeleton.
square.appendTo('#squareSkeleton');
`
```

Rectangle skeleton shape

```
`js
// Initialize Skeleton control for rectangle shape
var rect = new ej.notifications.Skeleton({
  shape: "Rectangle",
  height: '50px'
});
// Render initialized Skeleton.
rect.appendTo('#rectSkeleton');
```

Text skeleton shape

`js

// Initialize Skeleton control for text shape

```
var text = new ej.notifications.Skeleton({
  shape: "Text",
  height: "15px"
})
```

// Render initialized Skeleton.

text.appendTo('#textSkeleton');

Below example demonstrates the above functionalities of a Skeleton control.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize Skeleton control for circle shape
var cardProfile = new ej.notifications.Skeleton({shape: 'Circle', width:
'60px', height: '60px'});
// Render initialized Skeleton.
cardProfile.appendTo('#cardProfile');
var text1 = new ej.notifications.Skeleton({width: '30%', height: '15px'});
text1.appendTo('#text1');
var text2 = new ej.notifications.Skeleton({width: '15%', height: '15px'});
text2.appendTo('#text2');
var cardImage = new ej.notifications.Skeleton({shape: 'Rectangle', width:
'100%', height: '150px'});
cardImage.appendTo('#cardImage');
var rightOption = new ej.notifications.Skeleton({shape: 'Rectangle', width:
'20%', height: '32px'});
rightOption.appendTo('#rightOption');
var leftOption = new ej.notifications.Skeleton({shape: 'Rectangle', width:
'20%', height: '32px'});
leftOption.appendTo('#leftOption');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Skeleton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="skeletonCard" class="e-card" style="display: block">
            <div class="cardProfile">
                <div id="cardProfile"></div>
            </div>
            <div class="cardinfo">
                <div id="text1"></div><br>
                <div id="text2"></div>
            </div>
            <div class="cardContent">
                <div id="cardImage"></div>
            </div>
            <div class="cardoptions">
                <div id="rightOption"></div>
                <div id="leftOption"></div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Shimmer effect in EJ2 JavaScript Skeleton control

You can use the [shimmerEffect](#) property to change animation effect in the skeleton control. Skeleton supports **Wave**, **Pulse** and **Fade** effects and by default, the **shimmerEffect** is set to **Wave** effect.

```

`js
// Initialize Skeleton control with pulse effect
var pulse = ej.notifications.Skeleton({
    shape: 'Circle',
    width: "60px",
    shimmerEffect: 'Pulse'
});
// Render initialized Skeleton.
pulse.appendTo('#pulseSkeleton');
`

```


Below example demonstrates a list with pulse effect skeleton.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize Skeleton control
var listProfile = new ej.notifications.Skeleton({
  shape: 'Circle',
  width: '40px',
  height: '40px',
  shimmerEffect: 'Pulse'
});
// Render initialized Skeleton.
listProfile.appendTo('#listProfile');
var listProfile1 = new ej.notifications.Skeleton({
  shape: 'Circle',
  width: '40px',
  height: '40px',
  shimmerEffect: 'Pulse'
});
listProfile1.appendTo('#listProfile1');
var listCtn = new ej.notifications.Skeleton({
  width: '60%',
  height: '15px',
  shimmerEffect: 'Pulse'
});
listCtn.appendTo('#listCtn');
var listCtn1 = new ej.notifications.Skeleton({
  width: '60%',
  height: '15px',
  shimmerEffect: 'Pulse'
});
listCtn1.appendTo('#listCtn1');
var distCtn = new ej.notifications.Skeleton({
  width: '40%',
  height: '15px',
  shimmerEffect: 'Pulse'
});
distCtn.appendTo('#distCtn');
var distCtn1 = new ej.notifications.Skeleton({
  width: '40%',
  height: '15px',
  shimmerEffect: 'Pulse'
});
distCtn1.appendTo('#distCtn1');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Skeleton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <ul id="skeleton-list" class="e-card">
            <li>
                <div id="listProfile"></div>
                <div>
                    <div id="listCtn"></div><br>
                    <div id="distCtn"></div>
                </div>
            </li>
            <li>
                <div id="listProfile1"></div>
                <div>
                    <div id="listCtn1"></div><br>
                    <div id="distCtn1"></div>
                </div>
            </li>
        </ul>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Styles in EJ2 JavaScript Skeleton control

You can customize skeleton control in the below ways.

cssClass

You can customize the style of a Skeleton control by using [cssClass](#). The appearance of JavaScript Skeleton can be customized by changing the wave color, background color, width, and height. For detailed information, refer `styles.css` file below.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize Skeleton control
var circle = new ej.notifications.Skeleton({
    shape: 'Circle',
    width: "60px",
    cssClass: 'e-customize'

```

```
});
// Render initialized Skeleton.
circle.appendTo('#circleSkeleton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Skeleton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
notifications/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div style="position:relative;min-height:310px;border:1px solid;
padding:20px;">
      <div id="circleSkeleton"></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Visible

You can use the [visible](#) property which defines the visible state of Skeleton.

```
`js
```

```
// Initialize Skeleton control in hidden state
```

```
var circle: Skeleton = new ej.notifications.Skeleton({
shape: 'Circle',
width: "60px",
visible: false
});
```

```
// Render initialized Skeleton
circle.appendTo('#circleSkeleton');
`
```

Accessibility in EJ2 JavaScript Skeleton control

Accessibility is achieved in the Skeleton control through WAI-ARIA standard. The Skeleton control can be effectively accessed through assistive technologies such as screen readers.

ARIA attributes

The Skeleton control characterized with complete ARIA accessibility support that helps to be accessible by on-screen readers and other assistive technology devices. The following ARIA attributes are applicable for Skeleton control.

Properties	Functionality
role	This attribute is added to the input element to describe the actual role.
aria-label	Attribute provides the text label with some default description for the Skeleton.
aria-live	The aria-live attribute indicates the priority of updates to a live region.
aria-busy	This attribute is set to true when component is shown.

Smithchart

Getting started in EJ2 JavaScript Smithchart control

This section explains you the steps required to create a Smith Chart and demonstrate the basic usage of the Smith Chart control.

Dependencies

Below is the list of minimum dependencies required to use the Smith Chart.

```
`javascript
|-- @syncfusion/ej2-charts
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-svg-base
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-file-utils
`
```

Installation and Configuration

- To get you started with Smith Chart component, you can clone the

[Essential JS 2 quickstart](#) project

and install necessary packages by using the following commands.

`

```
git clone https://github.com/syncfusion/ej2-quickstart.git quickstart
```

```
cd quickstart
```

```
npm install
```

`

- **Smith Chart packages** need to be mapped in **system.config.js** configuration file.

```
`javascript
```

```
System.config({
```

```
paths: {
```

```
'syncfusion:': './node_modules/@syncfusion/',
```

```
},
```

```
map: {
```

```
app: 'app',
```

```
//Syncfusion packages mapping
```

```
"@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
```

```
"@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
```

```
"@syncfusion/ej2-svg-base": "syncfusion:ej2-svg-base/dist/ej2-svg-base.umd.min.js",
```

```
"@syncfusion/ej2-pdf-export": "syncfusion:ej2-pdf-export/dist/ej2-pdf-export.umd.min.js",
```

```
"@syncfusion/ej2-compression": "syncfusion:ej2-compression/dist/ej2-compression.umd.min.js",
```

```
"@syncfusion/ej2-file-utils": "syncfusion:ej2-file-utils/dist/ej2-file-utils.umd.min.js",
```

```
"@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
```

```
"@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
```

```
"@syncfusion/ej2-charts": "syncfusion:ej2-charts/dist/ej2-charts.umd.min.js",
```

```
},
```

```
packages: {
```

```
'app': { main: 'app', defaultExtension: 'js' }
```

```
}
```

```
});
```

```
`
```

The [project](#) is preconfigured with common settings (**src/styles/styles.css**, **system.config.js**) to start with all Essential JS 2 components.

Add Smith Chart to the Project

Add the HTML div element for Smith Chart into your `index.html`. [src/index.html]

```
`html
<!DOCTYPE html>
<html lang="en">
<head>
<title>EJ2 Animation</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Typescript UI Controls" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<script src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></script>
<script src="systemjs.config.js"></script>
</head>
<body>
<!--container which is going to render the Smith Chart-->
<div id='container'>
</div>
</body>
</html>
`
```

Now import the Smithchart component into your `index.ts` to instantiate a smithchart and append the smithchart instance to the `#container`

```
`javascript
import { Smithchart } from '@syncfusion/ej2-charts';
// initialize Smithchart component
let smithchart: Smithchart = new Smithchart();
// render initialized Smithchart
smithchart.appendTo('#container');
`
```

Now use the `npm run start` command to run the application in the browser.

npm run start

The below example shows a basic Smithchart.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart();
smithchart.appendTo('#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>

  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Module Injection

Smithchart component are segregated into individual feature-wise modules. In order to use a particular feature, you need to inject its feature module using `Smithchart.Inject()` method. In the current application, we are going to modify the above basic smithchart to visualize transmission lines.

For this application we are going to use tooltip and legend feature of the smithchart. Please find relevant feature module name and description as follows.

- SmithchartLegend - Inject this provider to use legend feature.

- TooltipRender - Inject this provider to use tooltip feature.

Now import the above mentioned modules from smithchart package and inject it into the Smithchart component using `Smithchart.Inject` method.

```
`javascript
```

```
import { Smithchart, SmithchartLegend, TooltipRender } from '@syncfusion/ej2-charts';
```

```
Smithchart.Inject(SmithchartLegend, TooltipRender);
```

```
,
```

Add Series to Smithchart

Smithchart had two type of specification for adding series.

- dataSource - Using this, Data object can bind directly by specifying the resistance and reactance values, series add to smithchart.
- points - Using this, collection of resistance and reactance values can bind directly to render series.

Below sample demonstrate adding two series to smithchart both ways.

- First series Transmission1 shows dataSource bound series.
- Second series Transmission2 shows points bound series.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  series: [
    {
      dataSource: [
        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
        { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
      ]
    }
  ]
});
```



```

    ],
    name: 'Transmission1',
    reactance: 'reactance', resistance: 'resistance'
  },
  {
    points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
    { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
    name: 'Transmission2'
  }
]
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add title to SmithChart

smithchart **title** API used to add title for smithchart. In that **text** API used to set text of the title.

API **visible** used to toggle the title.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    title: { text: 'Transmission lines applied for both impedance and admittance' },
    series: [
        {
            dataSource: [
                { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
                { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance: 0.2 },
                { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance: 0.5 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
                { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
                { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance: -1.0 }
            ],
            name: 'Transmission1',
            reactance: 'reactance', resistance: 'resistance'
        },
        {
            points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15 },
                { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance: 0.2 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 }
            ]
        }
    ]
});

```

```

        { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
        { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
        { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
        { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
    }],
    name: 'Transmission2'
  }
]
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable Marker to Smithchart

To use series marker and its customization in smithchart, use series **marker**. To display marker for particular series, need to specify **marker visible** as true. Below sample marker enabled for first series only.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  title: { text: 'Transmission lines applied for both impedance and admittance' },
  series: [
    {
      dataSource: [
        { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance: 0.2 },
        { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance: 0.5 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance: 0.0 },
        { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance: -1.0 }
      ],
      name: 'Transmission1',
      reactance: 'reactance', resistance: 'resistance',
      marker: {
        visible: true
      }
    },
    {
      points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15 },
        { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance: 0.2 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance: 0.2 },
        { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance: 0.8 },
        { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance: 1.6 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance: 1.6 },
        { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance: 4.5 },
        { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
      ],
    }
  ]
});

```

```

        name: 'Transmission2'
    }
    ]
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

    <title>Essential JS 2 for Smith chart </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
charts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

Enable DataLabel to Smithchart Marker

To use marker dataLabel and its customization in smithchart, use marker dataLabel. To display dataLabel for particular series marker, need to specify dataLabel visible as true in that series marker. Below sample dataLabel enabled for first series.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    title: { text: 'Transmission lines applied for both impedance and
admittance'},
    series: [
        {
            dataSource: [
```

```

    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
    { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
    { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
    ],
    name: 'Transmission1',
    reactance: 'reactance', resistance: 'resistance',
    marker: {
        visible: true,
        dataLabel: {
            visible: true
        }
    }
},
{
    points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
    { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
    name: 'Transmission2'
}
]
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable Legend for Smithchart

Smithchart had a legend feature, which is used to denote the correspond series. To enable the legend, need to inject `SmithchartLegend` module using `Smithchart.Inject(SmithchartLegend)` method and smithchart `legendSettings` visible as true. Following example sample shows enabling legend for smithchart. Series name can customize using series `name`.

INDEX.TS

```

import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
  title: { text: 'Transmission lines applied for both impedance and
admittance' },
  legendSettings: {
    visible: true
  },
  series: [
    {
      dataSource: [

```

```

        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
        { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
        { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
    ],
    name: 'Transmission1',
    reactance: 'reactance', resistance: 'resistance',
    marker: {
        visible: true,
        dataLabel: {
            visible: true
        }
    }
},
{
    points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
        { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
        { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
        { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
        { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
        { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
    }],
    name: 'Transmission2'
}
]
});
smithchart.appendTo('#container');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable Tooltip for Smithchart Series

Smithchart had a tooltip feature, which is used to show the current point's values. To enable the tooltip, need to inject **TooltipRender** module using **Smithchart.Inject(TooltipRender)** method and smithchart series **tooltip visible** as true. Following example sample shows enabling tooltip for smithchart series collection.

INDEX.TS

```

import { Smithchart, SmithchartLegend, TooltipRender} from '@syncfusion/ej2-
charts';
Smithchart.Inject(SmithchartLegend, TooltipRender);
let smithchart: Smithchart = new Smithchart({
  title: { text: 'Transmission lines applied for both impedance and
admittance'},
  legendSettings: {
    visible: true
  },
  series: [
    {
      dataSource: [

```

```

    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
    { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
    { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
    ],
    name: 'Transmission1',
    reactance: 'reactance', resistance: 'resistance',
    tooltip: {
        visible: true
    },
    marker: {
        visible: true,
        dataLabel: {
            visible: true
        }
    }
},
{
    points: [{ resistance: 0, reactance: 0.15 }, { resistance: 0,
reactance: 0.15 },
    { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
}],
    name: 'Transmission2',
    tooltip: {
        visible: true
    }
}

```

```

    }
  }
];
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Working with data in EJ2 JavaScript Smithchart control

Smithchart can visualise the data bound from local data. The data you bind for the smithchart, should be an array of object and that should contain the field resistance and reactance. This should be bind to points or datasource in the smithchart.

Data Binding

You can bind simple JSON data to smithchart using point property in series. JSON data should contain [resistance] and [reactance] fields. This JSON data should be bind to points or datasource in the smithchart. You can any number of JSON for points or datasource as per your requirement.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  series: [
```

```

    {
        points: [
            { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
            { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
            { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
        ],
    }, {
        points: [
            { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
            { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
            { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
            { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
            { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
            { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
            { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ],
    },
],
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Smith chart dimensions in EJ2 JavaScript Smithchart control

You can render the smithchart either corresponding to its container size or you can set the size of the smithchart as per your requirement. To render the smithchart corresponding to its container size, you need to set the size for the smithchart container. Else to set the size for the smithchart as per your requirement, you can use the width and height properties in the smithchart.

Size for Container

You can render smithchart to it's container size. To achieve this, you need to specify the width and height of the smithchart's container via inline or CSS as demonstrated below.

```

`javascript
<div id='container'>
<div id='element' style="width:650px; height:350px;"></div>
</div>
`

```

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },

```

```

    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ],
    },
  ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Size for Smithchart

You can also set size for smithchart directly through `[width]` and `[height]` properties. Using this properties, you can directly mention the width and height of the smithchart in pixels or you can set the width and height in percentage.

In Pixel

In smithchart's width and height property, you can directly give values in pixels like below demonstration. This will render smithchart in same size as you mentioned in you code.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  width: '650px',
  height: '300px',
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
    }
  ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In percentage

You can also specify the width and height of the smithchart in percentage. If you mention the width and height in percentage, then smithchart will be render as per the percentage of it's container size. You can set the values in percentage like below demonstration.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    width: '90%',
    height: '85%',
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ]
        }
    ]
});

```



```

    ],
  },
  ],
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Title subtitle in EJ2 JavaScript Smithchart control

Enable title

Title and subtitle is used to depicts the information about the data plotted in the smithchart. You can set the title and subtitle of the smithchart using the `[text]` property in title and subtitle. By default visibility

of the title as well as subtitle is enabled. You need to set simply text for title and subtitle in your sample as like below.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  title : {
    text: 'Impedance Transmission',
    subtitle: {
      text: 'Transmission'
    }
  },
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
    }
  ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
```

```

</div>
<script src="app.js"></script>
<script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
        loader.style.display = "none";
    }
    if (container) {
        container.style.visibility = "visible";
    }
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title trim

Both title and subtitle of the smithchart can be trimmed if it exceeds the certain length. Trimming is enabled using [enableTrim] for title as well as subtitle. This length can be changed using the property [maximumWidth]. Also [font], [textAlignment] and [visibility] can be customized for title as well as subtitle.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    title : {
        enableTrim: true,
        maximumWidth: 70,
        textAlignment: 'Center',
        visible: true,
        text: 'Impedance Transmission',
        subtitle: {
            text: 'Transmission',
            visible: true,
            textAlignment: 'Far',
            enableTrim: true,
            maximumWidth: 40
        }
    },
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },

```

```

        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ],
    },
],
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Smith chart axis in EJ2 JavaScript Smithchart control

Like chart, Smithchart is having support for two types of axis.

- Horizontal axis - axis drawn as straight line in the horizontal direction of the chart.
- Radial axis - axis is drawn as circular path.

Labels Customization

Axis labels are used to denote what kind of data is bound for smithchart. Using axis labels, you can easily identify in which interval chart is rendered. Using following properties we can customize the axis labels for horizontal and radial axis.

- `[labelPosition]` - used to place the labels either inside or outside the axis line.
- `[labelIntersectAction]` - used to hide the labels when intersect with other one.
- `[labelStyle]` - used to customize the properties such as font size, family, weight, opacity.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  radialAxis : {
    labelPosition: 'Inside',
    labelIntersectAction: 'Hide',
    labelStyle: {
      fontFamily: 'Times New Roman',
      fontWeight: 'bold',
      fontStyle: 'Italic',
      opacity: 0.75,
      size: '14px'
    }
  },
  horizontalAxis: {
    labelPosition: 'Inside',
    labelIntersectAction: 'Hide',
    labelStyle: {
      fontFamily: 'Times New Roman',
      fontWeight: 'bold',
      fontStyle: 'Italic',
      opacity: 0.75,
      size: '14px'
    }
  },
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
```

```

        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
        ],
    }, {
        points: [
            { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
            { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
            { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
            { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
            { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
            { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
            { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ],
    },
],
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Gridlines

To make the data in a chart that displays axes easier to read, you can display horizontal and radial axis gridlines. Gridlines extend from any horizontal and radial axes across the plot area of the smithchart.

Both horizontal and radial axis are having support for major as well as minor gridlines. Major gridlines are drawn from the position in which labels are rendered. Minor gridlines are drawn between two major gridlines as per the count we set in settings.

We can customize following things, in major as well as minor gridlines.

- **[width]** - used to customize the width of gridlines.
- **[dashArray]** - used to customize whether gridline have to render as normal line or dashed line.
- **[visible]** - used to enable or disable the visibility of the gridlines.
- **[opacity]** - used to customize the opacity of the major gridlines.
- **[count]** - used to customize the count of the minor gridlines.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  horizontalAxis: {
    majorGridLines: {
      visible: true,
      opacity: 0.8,
      width: 10
    }
  },
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
    }, {
      points: [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },

```

```

    { resistance: 9, reactance: -4.5 }, { resistance: 8,
    reactance: -3.5 },
    { resistance: 7, reactance: -2.5 }, { resistance: 6,
    reactance: -1.5 },
    { resistance: 5, reactance: -1 }, { resistance: 4.5,
    reactance: -0.5 },
    { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
    reactance: 0.4 },
    { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
    reactance: 0.5 },
    { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
    reactance: 0.2 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
    reactance: 0.05 },
    ],
    },
  ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Axisline

As name suggests that, it is a line in smithchart that can be configured to denotes the axis. By default, visibility of the axis line is true. You can customize its visibility by using visible property in axis Line. Other than visibility of the axis line, you can customize the following properties of the axis line.

- **[width]** - used to customize the width of the axis line.
- **[dashArray]** - used to render the axis line as dashed line.
- **[visible]** - used to enable or disable the visibility of the axis line.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  horizontalAxis: {
    majorGridLines: {
      visible: true,
      opacity: 0.8,
      width: 10
    },
    axisLine: {
      width: 10,
      visible: false
    }
  },
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
    }, {
      points: [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
      ],
    }
  ]
});
```

```

        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ],
    },
    ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

<!-- markdownlint-disable MD036 -->

Smith chart legend in EJ2 JavaScript Smithchart control

Legend is a key used in smithchart, that contains symbol and descriptions. It provides valuable information for interpreting what the smithchart is displaying and can be represented in various colors, shapes or other identifiers based on the data. In simple words, we can define that legend is used to denote the series rendered in the smithchart.

Position and Alignment

By default visibility of the legend is false. To enable the legend, kindly set visible as true in legendSettings. Default position for the legend is bottom. By using [position] property, you can change

the position of the legend. You can either place the legend at bottom, top, right and left side of the smithchart. To use the legend in smithchart, you need to import and inject the SmithchartLegend from chart.

INDEX.TS

```
import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
  legendSettings: {
    visible: true,
    position: 'Top'
  },
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
      name: 'Transmission1'
    }, {
      points: [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
      ],
      name: 'Transmission2'
    },
  ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Other than these positions, you can place the legend anywhere in the smithchart. To achieve this, you have to set position as custom in legendSettings and specify the x and y coordinates using the x and y properties in the location.

INDEX.TS

```

import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
  legendSettings: {
    visible: true,
    position: 'Custom',
    location: {
      x: 80,
      y: 100
    }
  }
});

```

```

    }
    },
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1'
        },
    ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
```

```

    }
    </script>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Alignment

Other than positioning the legend in the smithchart, you can customize its alignment also. By default, legend is aligned in center position. Using the `[alignment]` property, you can align the legend in near and far locations of the smithchart.

INDEX.TS

```

import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
    legendSettings: {
        visible: true,
        position: 'Top',
        alignment: 'Near'
    },
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1'
        }, {
            points: [
                { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
            ],
        }
    ]
});

```

```

        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ],
        name: 'Transmission2'
    },
    ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

Legend Shape

By default, legend is rendered in the circle shape and the color of the shape is as same as series color in the smithchart. Using the property `[shape]` in legend settings, you can change the icon shape of the legend as rectangle, triangle and so on.

INDEX.TS

```
import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
  legendSettings: {
    visible: true,
    position: 'Top',
    shape: 'Rectangle'
  },
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
      name: 'Transmission1'
    }, {
      points: [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
      ],
      name: 'Transmission2'
    },
  ],
});
```



```
smithchart.appendTo('#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Legend Size

By default, legend takes 20% - 25% of the chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the width vertically, while placing on left or right position of the chart. You can change this default legend size by using the [width] and [height] property of the legendSettings.

INDEX.TS

```
import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
```

```

    legendSettings: {
        visible: true,
        position: 'Top',
        height: 100,
        width: 200
    },
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1'
        }, {
            points: [
                { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
                { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
                { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
                { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
                { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
                { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
                { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
                { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
            ],
            name: 'Transmission2'
        },
    ],
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Smith chart </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Smithchart UI
Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Padding

You can customize the space between two legend items and space between legend shape and text as per your requirement. For customizing the space between two legend items, you can use `[itemPadding]` property. To control space between legend shape and text, you can use `[shapePadding]` property.

INDEX.TS

```

import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
    legendSettings: {
        visible: true,
        position: 'Top',
        itemPadding: 5,
        shapePadding: 10
    },
    series: [
        {
            points: [

```

```

    { resistance: 10, reactance: 25 }, { resistance: 8,
    reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
    reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
    reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
    reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
    reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
    reactance: 0.15 },
    ],
    name: 'Transmission1'
  }, {
    points: [
      { resistance: 20, reactance: -50 }, { resistance: 10,
    reactance: -10 },
      { resistance: 9, reactance: -4.5 }, { resistance: 8,
    reactance: -3.5 },
      { resistance: 7, reactance: -2.5 }, { resistance: 6,
    reactance: -1.5 },
      { resistance: 5, reactance: -1 }, { resistance: 4.5,
    reactance: -0.5 },
      { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
    reactance: 0.4 },
      { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
    reactance: 0.5 },
      { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
    reactance: 0.2 },
      { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
    reactance: 0.05 },
    ],
    name: 'Transmission2'
  },
],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Toggle Visibility

By default series name is displayed in the legend. You can collapse the visibility of the series by clicking the legend for the particular series. You can toggle the series visibility as true or false using the [toggleVisibility] property. By default it is true.

INDEX.TS

```

import { Smithchart, SmithchartLegend } from '@syncfusion/ej2-charts';
Smithchart.Inject(SmithchartLegend);
let smithchart: Smithchart = new Smithchart({
    legendSettings: {
        visible: true,
        position: 'Top',
        toggleVisibility: false
    },
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            ]
        }
    ]
});

```

```

        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
        ],
        name: 'Transmission1'
    }, {
        points: [
            { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
            { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
            { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
            { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
            { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
            { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
            { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ],
        name: 'Transmission2'
    },
    ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Smith chart </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
```

```

        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    }
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Smith chart tooltip in EJ2 JavaScript Smithchart control

Smithchart will display details about the points through tooltip, when the mouse is moved over the point. By default, tooltip is disabled. To enable the tooltip for smithchart, you need to import and inject TooltipRender module from chart. And also set the property visible as true, in tooltip settings. You can customize the tooltip's visibility and appearance differently each series in the smithchart.

INDEX.TS

```

import { Smithchart, TooltipRender } from '@syncfusion/ej2-charts';
Smithchart.Inject(TooltipRender);
let smithchart: Smithchart = new Smithchart({
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1',
            tooltip: {
                visible: true
            },
            marker: {
                visible: true
            }
        }, {
            points: [
                { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
            ],
        }
    ]
});

```

```

        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ],
    name: 'Transmission2',
    tooltip: {
        visible: true
    },
    marker: {
        visible: true
    }
},
],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
  </script>
</body>
```



```

    }
    if (container) {
        container.style.visibility = "visible";
    }
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Smith chart marker in EJ2 JavaScript Smithchart control

Markers and Datalabels are used to provide information about the data points in the series. You can add a shape to adorn each data point. By default marker and datalabel both are disabled in smithchart. You can enable both of them by setting visible property as true in marker and datalabel settings

Marker

Default visibility of marker is false. You can enable the marker by setting property visible as true in marker settings. This will add marker for each point in the series. Using marker setting, you can customize marker differently for each series in smithchart.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1',
            marker: {
                visible: true
            }
        }
    ],
});
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Marker Customization

Using marker settings in series, you can customize the marker for each series differently. Using marker settings, you can customize following properties differently for each series in the smithchart.

- [width] - To control the width of the marker.
- [height] - To control the height of the marker.
- [fill] - Used to customize the fill color of the marker.
- [opacity] - Used to customize the opacity of the marker.
- [border] - Used to control the width and color of the marker's border.
- [shape] - Used to change the shape of the marker.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
      name: 'Transmission1',
      marker: {
        visible: true,
        height: 10,
        width: 10,
        fill: '#ff99ff',
        opacity: 1,
        shape: 'rectangle',
        border: {
          color: '#cc00cc',
          width: 2
        }
      }
    }
  ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Datalabels

By default, datalabel is disabled. You can enable the datalabel by setting property visible as true in datalabel settings. For each point in series, data label is created. Datalabel for each series can be customized differently using datalabel settings.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1',
            marker: {
                dataLabel: {
                    visible: true
                }
            }
        }
    ]
});

```

```

    }
    ],
  });
smithchart.appendTo('#container');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Datalabel customization

Using datalabel settings in marker, you can customize the datalabel for each series differently. In datalabel, you can customize the following properties differently for each series.

- **[fill]** - Used to changes the fill color of the data label's shape.

- **[opacity]** - Used to control the opacity of the data label's shape.
- **[border]** - Used to customize the width and color of the border.
- **[textStyle]** - Used to customize the font color, width and size.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
                { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
                { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ],
            name: 'Transmission1',
            marker: {
                dataLabel: {
                    visible: true,
                    fill: '#99ffcc',
                    opacity: 1,
                    border: {
                        color: '#1aff8c',
                        width: 2,
                    }
                }
            }
        }
    ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Smith chart </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Smith chart series in EJ2 JavaScript Smithchart control

You can add any number of series to the smithchart as per your requirement. You can use series setting to either add or customize the data. For the points or datasource added in the series, line is drawn. You can customize the each series as per your requirement with marker, datalabel, animation, opacity and so on.

points or datasource

For adding values in the smithchart, you can use either points or datasource in the series. Points and datasource both should be array of object which should contain the field names resistance and reactance.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },

```

```

        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ]
    },
    ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
  </div>
  <script src="app.js"></script>
  <script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
      loader.style.display = "none";
    }
    if (container) {
      container.style.visibility = "visible";
    }
  </script>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```


Series customization

Using following options in series settings, you can customize each series in smithchart as per your requirement.

- **[fill]** - Used to customize the fill color for the series.
- **[enableSmartLabels]** - Used to place the data labels on the smithchart without overlapping with each other.
- **[visibility]** - Used to handle the visibility of the series.
- **[opacity]** - Used to control the opacity of the series line.
- **[width]** - Used to customize the width of the series line.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
      name: 'Transmission1',
      fill: '#009933',
      visibility: 'visible',
      opacity: 0.75,
      width: 2.5,
      marker: {
        dataLabel: {
          visible: true
        }
      }
    }
  ],
});
smithchart.appendTo('#container');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
    </div>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Smith chart print in EJ2 JavaScript Smithchart control

Print

The rendered smithchart can be printed directly from the browser by calling the public method print. ID of the smithchart's div element must be passed as argument to that method.

INDEX.TS

```

import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
    series: [
        {
            points: [
                { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
                { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
                { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
                { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            ]
        }
    ]
});

```

```

        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
        ],
    }
],
});
smithchart.appendTo('#element');
document.getElementById('print').onclick = () => {
    smithchart.print();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Smith chart </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container"></div>
    <div id="element" style="float: left "></div>
    <button id="print" type="button" width="15%" style="float:
right">Print</button>
    <script src="app.js"></script>
    <script>
        let loader = document.getElementById('loader');
        let container = document.getElementById('container');
        if (loader) {
            loader.style.display = "none";
        }
        if (container) {
            container.style.visibility = "visible";
        }
    </script>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

    </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Export

The rendered smithchart can be exported to JPEG , PNG, SVG or PDF format by using export method in smithchart. Input parameters for this method are Export Type for format and fileName of result.

INDEX.TS

```
import { Smithchart } from '@syncfusion/ej2-charts';
let smithchart: Smithchart = new Smithchart({
  series: [
    {
      points: [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
      ],
    }
  ],
});
smithchart.appendTo('#element');
document.getElementById('print').onclick = () => {
  smithchart.export('PNG', 'result');
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Smith chart </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Smithchart UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container"></div>
<div id="element" style="float: left "></div>
<button id="print" type="button" width="15%" style="float:
right">Print</button>
<script src="app.js"></script>
<script>
    let loader = document.getElementById('loader');
    let container = document.getElementById('container');
    if (loader) {
        loader.style.display = "none";
    }
    if (container) {
        container.style.visibility = "visible";
    }
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Smithchart control

The Smith chart control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Smith chart control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

```

WAI-ARIA attributes

The Smith chart control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Smith chart control:

- `img` (role)
- `region` (role)
- `aria-label` (attribute)
- `aria-hidden` (attribute)

Keyboard interaction

The Smith chart control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Smith chart control.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Moves the focus to the next element in the Smith chart. |

| **Shift + Tab** | Moves the focus to the previous element in the Smith chart. |

| **Ctrl + P** | Prints the Smith chart. |

Ensuring accessibility

The Smith chart control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Smith chart control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Smith chart control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

Sparkline

Sparkline dimensions in EJ2 JavaScript Sparkline control

Size for container

Sparkline can be rendered to its container size. You can set the size through inline or CSS as shown in the following code.

```
,
<div id='container'>
<div id='element' style="width:650px; height:350px;"></div>
</div>
,
```

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  dataSource: [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ],
  // Assign the dataSource values to series of sparkline 'xName and yName'
  xName: 'xval', yName: 'yval'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Size for sparkline

You can also set the size for sparkline directly using the [width](#) and [height](#) properties.

In pixel

You can set the size for sparkline in pixel as demonstrated in the following code.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  width: '350', height: '150',
  dataSource: [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ],
  // Assign the dataSource values to series of sparkline 'xName and
  yName'
  xName: 'xval', yName: 'yval'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
  Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  charts/styles/material.css" rel="stylesheet">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In percentage

By setting values in percentage, sparkline gets its dimension with respect to its container. For example, when the height is set to '50%', sparkline is rendered to half of its container height.

INDEX.TS

```

import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
let sparkline: Sparkline = new Sparkline({
    width: '80%', height: '50%',
    dataSource: [
        { x: 0, xval: '2005', yval: 20090440 },
        { x: 1, xval: '2006', yval: 20264080 },
        { x: 2, xval: '2007', yval: 20434180 },
        { x: 3, xval: '2008', yval: 21007310 },
        { x: 4, xval: '2009', yval: 21262640 },
        { x: 5, xval: '2010', yval: 21515750 },
        { x: 6, xval: '2011', yval: 21766710 },
        { x: 7, xval: '2012', yval: 22015580 },
        { x: 8, xval: '2013', yval: 22262500 },
        { x: 9, xval: '2014', yval: 22507620 },
    ],
    // Assign the dataSource values to series of sparkline 'xName and
    yName'
    xName: 'xval', yName: 'yval'
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Essential JS 2 for Sparkline UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sparkline types in EJ2 JavaScript Sparkline control

Different types of shapes can be used to represent the sparkline. You can change the sparkline type by setting the type property. Sparkline supports the following types:

- Line
- Column
- Win-Loss
- Pie
- Area

The following code sample shows different types of sparklines.

Line

The [Line] type is used to render the sparkline series as line.

INDEX.TS

```

import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
let sparkline: Sparkline = new Sparkline({
    height: '100px',
    width: '70%',
    dataSource: [
        { x: 0, xval: '2005', yval: 20090440 },
        { x: 1, xval: '2006', yval: 20264080 },
        { x: 2, xval: '2007', yval: 20434180 },
        { x: 3, xval: '2008', yval: 21007310 },
    ]
});

```

```

        { x: 4, xval: '2009', yval: 21262640 },
        { x: 5, xval: '2010', yval: 21515750 },
        { x: 6, xval: '2011', yval: 21766710 },
        { x: 7, xval: '2012', yval: 22015580 },
        { x: 8, xval: '2013', yval: 22262500 },
        { x: 9, xval: '2014', yval: 22507620 },
    ],
    // Assign the dataSource values to series of sparkline 'xName and yName'
    xName: 'xval', yName: 'yval',
    // Assign 'Line' as type of the sparkline
    type: 'Line'
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Column

The [Column] type is used to render the sparkline series as column.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '150px',

```

```

width: '200px',
dataSource: [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
],
// Assign the dataSource values to series of sparkline 'xName and yName'
xName: 'xval', yName: 'yval',
// Assign 'Column' as type of the sparkline
type: 'Column'
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Pie

The [Pie] type is used to render the sparkline series as pie.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '70%',
    dataSource: [
        { x: 0, xval: '2005', yval: 20090440 },
        { x: 1, xval: '2006', yval: 20264080 },
        { x: 2, xval: '2007', yval: 20434180 },
        { x: 3, xval: '2008', yval: 21007310 },
        { x: 4, xval: '2009', yval: 21262640 },
        { x: 5, xval: '2010', yval: 21515750 },
        { x: 6, xval: '2011', yval: 21766710 },
        { x: 7, xval: '2012', yval: 22015580 },
        { x: 8, xval: '2013', yval: 22262500 },
        { x: 9, xval: '2014', yval: 22507620 },
    ],
    // Assign the dataSource values to series of sparkline 'xName and yName'
    xName: 'xval', yName: 'yval',
    // Assign 'Pie' as type of the sparkline
    type: 'Pie'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

WinLoss

The [WinLoss] type is used to render the sparkline series as WinLoss.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '100px',
  width: '70%',
  dataSource: [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: -20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: -21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: -22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ],
  // Assign the dataSource values to series of sparkline 'xName and yName'
  xName: 'xval', yName: 'yval',
  // Assign 'WinLoss' as type of the sparkline
  type: 'WinLoss'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Area

The [Area] type is used to render the sparkline series as area.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '100px',
  width: '70%',
  dataSource: [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ],
  // Assign the dataSource values to series of sparkline 'xName and yName'
  xName: 'xval', yName: 'yval',
  // Assign 'Area' as type of the sparkline
  type: 'Area'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis customization in EJ2 JavaScript Sparkline control

You can customize axis value types and min and max values of the sparkline.

Change value type of the sparkline

You can change the sparkline value type by setting the `[valueType]` property to `[Numeric]`, `[Category]`, or `[DateTime]`.

DateTime

You can assign date-time values to the sparkline by setting the `[valueType]` property to `[DateTime]`.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '150px',
    width: '130px',
    dataSource: [
        { xDate: new Date(2018, 0, 1), x: 0, yval: 4 },
        { xDate: new Date(2018, 0, 2), x: 1, yval: 4.5 },
        { xDate: new Date(2018, 0, 3), x: 2, yval: 8 },
        { xDate: new Date(2018, 0, 4), x: 3, yval: 7 },
        { xDate: new Date(2018, 0, 5), x: 4, yval: 6 },
        { xDate: new Date(2018, 0, 8), x: 5, yval: 8 },
        { xDate: new Date(2018, 0, 9), x: 6, yval: 8 },
        { xDate: new Date(2018, 0, 10), x: 7, yval: 6.5 },
        { xDate: new Date(2018, 0, 11), x: 8, yval: 4 },
        { xDate: new Date(2018, 0, 12), x: 9, yval: 5.5 }
    ],
    // Assign the dataSource values to series of sparkline 'xName and yName'
    xName: 'xDate', yName: 'yval',
    // Assign the "Column" as type of the sparkline
    type: 'Column',
    // Assign the "DateTime" as value type to the sparkline
    valueType: 'DateTime'
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```



```

<title>Essential JS 2 for Sparkline </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for Sparkline UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Category

You can assign category values to the sparkline by setting [valueType] to [Category].

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '150px',
    width: '130px',
    dataSource: [
        { x: 0, xval: 'Robert', yval: 60 },
        { x: 1, xval: 'Andrew', yval: 65 },
        { x: 2, xval: 'Suyama', yval: 70 },
        { x: 3, xval: 'Michael', yval: 80 },
        { x: 4, xval: 'Janet', yval: 55 },
        { x: 5, xval: 'Davolio', yval: 90 },
        { x: 6, xval: 'Fuller', yval: 75 },
        { x: 7, xval: 'Nancy', yval: 85 }
    ],
    // Assign the dataSource values to series of sparkline 'xName and yName'
    xName: 'xval', yName: 'yval',
    // Assign the 'Column' as type of the sparkline
    type: 'Column',
    // Assign the "Category" as value type of the sparkline
    valueType: 'Category'
});

```

```
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Numeric

You can assign numeric values to the sparkline by setting [valueType] to [Numeric].

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '150px',
  width: '130px',
  dataSource: [
    { x: 1, xval: 2010, yval: 190 },
    { x: 2, xval: 2011, yval: 165 },
    { x: 3, xval: 2012, yval: 158 },
    { x: 4, xval: 2013, yval: 175 },
    { x: 5, xval: 2014, yval: 200 },
    { x: 6, xval: 2015, yval: 180 },
    { x: 7, xval: 2016, yval: 210 }
  ],
  // Assign the dataSource values to series of sparkline 'xName and yName'
  xName: 'x', yName: 'yval',
```

```
// Assign the 'Column' as type of the sparkline
type: 'Column',
// Assign the 'Numeric' as value type of the sparkline
valueType: 'Numeric'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Change min and max values of axis

You can change the min and max values of x-axis by setting the [minX] and [maxX] values to the [axisSettings] property. You can also change the min and max values of y-axis by setting the [minY] and [maxY] values to the [axisSettings] property.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '150px',
  width: '130px',
  dataSource: [
    { x: 0, yval: 50 },
    { x: 1, yval: 30 },
    { x: 2, yval: 20 },
```

```

        { x: 3, yval: 30 },
        { x: 4, yval: 50 },
        { x: 5, yval: 40 },
        { x: 6, yval: 20 },
        { x: 7, yval: 10 },
        { x: 8, yval: 30 },
        { x: 9, yval: 10 },
        { x: 10, yval: 40 }
    ],
    // Assign the dataSource values to series of sparkline 'xName and yName'
    xName: 'x', yName: 'yval',
    // Set min and max values to the y-axis of sparkline
    axisSettings: {
        minY: 0, maxY: 150
    },
    // Assign the 'Column' as type of the sparkline
    type: 'Column',
    // Assign the 'Numeric' as value type of the sparkline
    valueType: 'Numeric'
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Change value of axis

You can set horizontal axis line value of the sparkline by setting [value] to the [axisSettings] property. The following code example shows this.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '150px',
    width: '130px',
    dataSource: [
        { x: 0, yval: 50 },
        { x: 1, yval: 30 },
        { x: 2, yval: 20 },
        { x: 3, yval: 30 },
        { x: 4, yval: 50 },
        { x: 5, yval: 40 },
        { x: 6, yval: 20 },
        { x: 7, yval: 10 },
        { x: 8, yval: 30 },
        { x: 9, yval: 10 },
        { x: 10, yval: 40 }
    ],
    // Assign the dataSource values to series of sparkline 'xName and yName'
    xName: 'x', yName: 'yval',
    // Set the value to horizontal axis of sparkline
    axisSettings: {
        value: 25
    },
    // Assign the 'Column' as type of the sparkline
    type: 'Column',
    // Assign the 'Numeric' as value type of the sparkline
    valueType: 'Numeric'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

    <div id="container" style="margin-top: 15%;">
      <div id="element" align="center"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Axis line customization

Axis of the sparkline can be collapsed using the `visible` property in `lineSettings`; this is not applicable for win-loss. You can customize the `color`, `width`, `opacity`, and `dashArray` of axis line.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '150px',
  width: '130px',
  dataSource: [
    { x: 0, yval: 50 },
    { x: 1, yval: 30 },
    { x: 2, yval: 20 },
    { x: 3, yval: 30 },
    { x: 4, yval: 50 },
    { x: 5, yval: 40 },
    { x: 6, yval: 20 },
    { x: 7, yval: 10 },
    { x: 8, yval: 30 },
    { x: 9, yval: 10 },
    { x: 10, yval: 40 }
  ],
  // Assign the dataSource values to series of sparkline 'xName and yName'
  xName: 'x', yName: 'yval',
  // Set the value to horizontal axis
  axisSettings: {
    // To configure axis line settings
    lineSettings: {
      visible: true,
      color: "#ff14ae",
      dashArray: '5,3'
    }
  },
  // Assign the value type as 'Numeric' to the sparkline
  valueType: 'Numeric'
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 for Sparkline </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for Sparkline UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Special points customization in EJ2 JavaScript Sparkline control

You can customize the points by initializing the point colors. The customization options allows to differentiate the [start], [end], [positive], [negative], and [low] points. This customization is only applicable for line, column, and area type sparklines.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '150px',
    width: '130px',
    dataSource: [
        { x: 0, xval: 'AUDI', yval: 1 },
        { x: 1, xval: 'BMW', yval: 5 },
        { x: 2, xval: 'BUICK', yval: -1 },
        { x: 3, xval: 'CETROEN', yval: -6 },
        { x: 4, xval: 'CHEVROLET', yval: 0 },
        { x: 5, xval: 'FIAT', yval: 1 },
        { x: 6, xval: 'FORD', yval: -2 },
        { x: 7, xval: 'HONDA', yval: 7 },
        { x: 8, xval: 'HYUNDAI', yval: -9 },
        { x: 9, xval: 'JEEP', yval: 0 },
        { x: 10, xval: 'KIA', yval: -10 },
        { x: 11, xval: 'MAZDA', yval: 3 }
    ],

```

```
// Assign the dataSource values to series of sparkline 'xName and
yName'
xName: 'xval', yName: 'yval',
// Assign 'Column' as type of the sparkline
type: 'Column',
// Assign "Category" as the value type of the sparkline
valueType: 'Category',
// To configure sparkline series highest y value point color.
highPointColor: 'blue',
// To configure sparkline series lowest y value point color.
lowPointColor: 'orange',
// To configure sparkline series first x value point color.
startPointColor: 'green',
// To configure sparkline series last x value point color.
endPointColor: 'green',
// To configure sparkline series negative y value point color.
negativePointColor: 'red'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Tie point color

Tie point color is used to configure the win-loss series type sparkline's y-value point color. The following code sample shows the tie point color of sparkline series.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '150px',
    width: '130px',
    dataSource: [12, 15, -10, 13, 15, 6, -12, 17, 13, 0, 8, -10],
    // Assign the 'WinLoss' as type of Sparkline
    type: 'WinLoss',
    // Assign "Numeric" as the value type of the sparkline
    valueType: 'Numeric',
    tiePointColor: 'blue'
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Range band in EJ2 JavaScript Sparkline control

This section explains how to customize the sparkline with multiple range bands.

Range band customization

The range band feature is used to highlight a particular range along with the y-axis using the [startRange] and [endRange] properties. You can also customize the [color] and [opacity] of the range band.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '150px',
    width: '150px',
    lineWidth: 2,
    fill: '#0d3c9b',
    dataSource: [0, 6, 4, 1, 3, 2, 5],
    // To configure range band settings
    rangeBandSettings: [
        {
            startRange: 1,
            endRange: 3,
            color: '#bfd4fc',
            opacity: 0.4
        }
    ]
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Multiple range band customization

You can define multiple range bands to a sparkline as shown in the following code sample.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '150px',
  width: '150px',
  lineWidth: 2,
  fill: '#0d3c9b',
  dataSource: [0, 6, 4, 1, 3, 2, 5],
  rangeBandSettings: [
    {
      startRange: 1,
      endRange: 2,
      color: '#bfd4fc',
      opacity: 0.4
    },
    {
      startRange: 4,
      endRange: 5,
      color: 'red',
      opacity: 0.4
    }
  ]
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Marker in EJ2 JavaScript Sparkline control

This section explains how to add markers to the sparklines.

Adding marker to the sparkline

To add marker to the sparkline, specify the **visible** of **markerSettings** as following values. The **visible** will accept multiple values too.

- All - Enables markers for all points.
- Start - Enables marker for the start point.
- End - Enables marker for the end point.
- High - Enables marker for the high point.
- Low - Enables marker for the low point.
- Negative - Enables markers for the negative points.

The following code example shows enabling markers for all points.

INDEX.TS

```
import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    },
    dataSource: [0, 6, 4, 1, 3, 2, 5],
    // To enable markers for all points
    markerSettings: {
        visible: ['All']
    }
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 for Sparkline </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for Sparkline UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Adding marker to special point

In sparkline, markers can be enabled for particular points such as the start, end, low, high, or negative. The following code examples shows enabling markers for the high and low points.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    },
    dataSource: [3, 6, 4, 1, 3, 2, 5],
    // To enable marker for high and low points with color customization
    highPointColor: 'blue',
    lowPointColor: 'red',
    markerSettings: {
        visible: ['High', 'Low']
    }
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
    <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customizing markers

Sparkline markers can be customized in terms of fill color, border color, width, opacity, and size. The following code example shows customizing marker's fill, border, and size.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    },
    dataSource: [3, 6, 4, 1, 3, 2, 5],
    // To enable marker for high and low points with color customization
    fill: 'blue',
    markerSettings: {
        visible: ['All'],
        size: 5, fill: 'white', border: { color: 'blue', width: 2}
    }
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Sparkline UI
Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div id="element" align="center"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data labels in EJ2 JavaScript Sparkline control

Data labels are used to display values of data points to improve the readability.

Enable data label

To enable data label for sparkline series, provide **visible** of the **dataLabelSettings** as following possible values:

- All - Enables data label of all points.
- Start - Enables data label of the start point.
- End - Enables data label of the end point.
- High - Enables data label of the high point.
- Low - Enables data label of the low point.
- Negative - Enables data labels of the negative points.

The following example shows enabling sparkline data label for all points.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '350px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    },
    fill: 'blue',
    dataSource: [0, 6, 4, 1, 3, 2, 5],
    // To enable data label for all points
    dataLabelSettings: {

```

```

        visible: ['All']
    },
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize data label

Data labels can be customized using the fill, border, opacity, and textStyle. The following code example shows customizing data label border, text color, and fill color.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '200px',
  width: '350px',
  axisSettings: {
    minX: -1, maxX: 7, maxY: 8, minY: -1
  },
  fill: 'blue',
  dataSource: [0, 6, 4, 1, 3, 2, 5],
  // To customize data label fill, border, and text color
  dataLabelSettings: {

```



```

        visible: ['All'],
        border: { color: 'blue', width: 1},
        fill: 'blue', opacity: 0.4,
        textStyle: {
            color: 'white'
        }
    },
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Format data label text

The text of data labels can be formatted using the `format` API in the sparkline `dataLabelSettings`. By default, data label shows the y-value of point. The following code example shows how to display x and y-values for points.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '200px',
  width: '500px',
  axisSettings: {

```

```

        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    fill: 'blue',
    valueType: 'Category',
    xName: 'x', yName: 'y',
    dataSource: [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed', y: 2 }, {x:
'Thu', y: 4 }, {x: 'Fri', y: 6 }, ],
    // To enable data label for all points
    dataLabelSettings: {
        format: '${x} : ${y}',
        visible: ['All'],
        border: { color: 'blue', width: 1},
        fill: 'blue', opacity: 0.4,
        textStyle: {
            color: 'white'
        }
    },
    },
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

User interaction in EJ2 JavaScript Sparkline control

Sparkline has two user interaction features: tooltip and tracker line.

Tooltip

The sparkline provides options to display details about values of data points through tooltips when hovering the mouse over data point. To use tooltip in sparkline, inject the `SparklineTooltip` module to sparkline using the inject method.

The following code example shows enabling tooltip for sparkline with format.

INDEX.TS

```
import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    fill: 'blue',
    valueType: 'Category',
    xName: 'x', yName: 'y',
    dataSource: [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed', y: 2 }, {x:
'Thu', y: 4 }, {x: 'Fri', y: 6 },],
    // To enable tooltip for sparkline
    tooltipSettings: {
        visible: true,
        // formatting tooltip text
        format: '${x} : ${y}'
    }
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div class="spark" id="element" align="center"></div>
    </div>
<style>
```

```

        .sparktooltip {
            border-radius: 5px;
            background: #008cff;
            color: #FFFFFF !important;
            font-size: 16px;
            font-style: italic;
            padding: 8px;
        }
    </style>
    <script>
    var ele = document.getElementById('container');
    if(ele) {
        ele.style.visibility = "visible";
    }
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip customization

The fill color, text styles, format, and border of the tooltip can be customized. The following code example shows customization of tooltip's fill color and text style.

INDEX.TS

```

import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    fill: '#033e96',
    valueType: 'Category',
    xName: 'x', yName: 'y',
    dataSource: [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed', y: 2 }, {x:
'Thu', y: 4 }, {x: 'Fri', y: 6 },],
    // To enable tooltip for sparkline with fill color customization
    tooltipSettings: {
        visible: true,
        format: '${x} : ${y}',
        fill: '#033e96',
        textStyle: {
            color: 'white'
        },
    },
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Essential JS 2 for Sparkline UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div class="spark" id="element" align="center"></div>
    </div>
<style>
    .sparktooltip {
        border-radius: 5px;
        background: #008cff;
        color: #FFFFFF !important;
        font-size: 16px;
        font-style: italic;
        padding: 8px;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip template

Sparkline tooltip has template support. By using tooltip template, you can customize tooltips. The following code example shows more customization options provided to `sparktooltip` css class that is used in tooltip template div. Using this template, images also can be added to tooltip.

```

.sparktooltip {
border-radius: 5px;
background: #008cff;
color: #FFFFFF !important;
font-size: 16px;
font-style: italic;
padding: 8px;

```

```

}
,

```

INDEX.TS

```

import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    },
    fill: '#033e96',
    valueType: 'Category',
    xName: 'x', yName: 'y',
    dataSource: [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed', y: 2 }, {x:
'Thu', y: 4 }, {x: 'Fri', y: 6 }, ],
    // To enable tooltip template for sparkline with fill color, border
    radius and padding customization
    tooltipSettings: {
        visible: true,
        template: '<div class="sparktooltip">${x} : ${y}<div>'
    }
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div class="spark" id="element" align="center"></div>
    </div>
<style>
    .sparktooltip {
        border-radius: 5px;
        background: #008cff;
        color: #FFFFFF !important;
    }

```

```

        font-size: 16px;
        font-style: italic;
        padding: 8px;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Track line

The track line tracks data points that are closer to the mouse position or touch contact.

To enable track lines for sparkline, specify the `visible` option of `trackLineSettings` to true. Based on theme, tracker color will be changed. The default value of tracker color is black.

To use track line in sparkline, inject the `SparklineTooltip` module to sparkline using the inject method.

INDEX.TS

```

import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
let sparkline: Sparkline = new Sparkline({
    height: '200px',
    width: '500px',
    axisSettings: {
        minX: -1, maxX: 46, maxY: 10, minY: -1
    },
    fill: '#033e96',
    dataSource: [5, 3, 4, 6, 8, 7, 9, 1, 3, 5, 3, 4, 6, 8, 7, 9, 1, 3, 5,
2, 4, 6, 7, 9, 5, 8, 3, 6, 1, 7, 4, 2, 5, 2, 4, 6, 7, 9, 5, 8, 3, 6, 1, 7,
4, 2],
    // To enable tooltip template for sparkline with fill color, border
    radius and padding customization
    tooltipSettings: {
        trackLineSettings: {
            visible: true,
            color: '#033e96',
            width: 1
        }
    }
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 for Sparkline </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Sparkline UI
Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" style="margin-top: 15%;">
        <div class="spark" id="element" align="center"></div>
    </div>
<style>
    .sparktooltip {
        border-radius: 5px;
        background: #008cff;
        color: #FFFFFF !important;
        font-size: 16px;
        font-style: italic;
        padding: 8px;
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Appearance in EJ2 JavaScript Sparkline control

The appearance of the sparkline can be customized using margin, containerArea border, and containerArea background.

Sparkline border

The **containerArea border** of the sparkline is used to render border to cover sparkline area.

The following code example shows the sparkline with overall border.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
    // To render border around the sparkline
    containerArea: {
        border: { color: '#033e96', width: 2 }
    },
    border: { color: '#033e96', width: 1 },
    height: '200px',
    width: '350px',
    type: 'Area',

```



```

    fill: '#b2cfff',
    dataSource: [3, 6, 4, 1, 3, 2, 5]
  });
  sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div class="spark" id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Sparkline padding

Padding is used to specify padding value between container and sparkline. By default, padding value of the sparkline is 5. Sparkline padding values are specified by the left, right, top, and bottom.

The following code example shows the sparkline with overall padding is set to 20.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  containerArea: {
    border: { color: '#033e96', width: 2 }
  },
  // To render sparkline with padding
  padding: { left: 20, right: 20, bottom: 20, top: 20 },
  border: { color: '#033e96', width: 1 },
  height: '200px',
  width: '350px',

```

```

    type: 'Area',
    fill: '#b2cfff',
    dataSource: [3, 6, 4, 1, 3, 2, 5]
  });
  sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div class="spark" id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Sparkline area customization

The background color of the sparkline area can be customized using the `containerArea` background color. By default, the sparkline background color is `transparent`.

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  containerArea: {
    border: { color: '#033e96', width: 2 },
    // To render sparkline with background
    background: '#eff1f4',
  },
  padding: { left: 20, right: 20, bottom: 20, top: 20 },
  border: { color: '#033e96', width: 2 },
  height: '200px',

```

```

width: '350px',
type: 'Area',
fill: '#b2cfff',
dataSource: [3, 6, 4, 1, 3, 2, 5]
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div class="spark" id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Sparkline theme

Datalabel and trackline colors of the sparkline will be changed based on theme. For example, for dark theme, the color of datalabel and trackline should be white; for light theme, their value should be black. The possible values for sparkline theme are **Material**, **Fabric**, **Bootstrap**, and **Highcontrast**.

The following code example shows the color for datalabel and trackline is set to white for dark theme.

INDEX.TS

```

import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
document.getElementById('container').style = 'background: #000000; margin-
top: 15%;';
let sparkline: Sparkline = new Sparkline({
  // To specify theme

```

```

theme: 'HighContrast',
dataLabelSettings: { visible: ['All'] },
tooltipSettings: {
  trackLineSettings: { visible: true }
},
axisSettings: {
  minX: -1, maxX: 7
},
lineWidth: 3,
border: { color: 'transparent', width: 2 },
height: '200px',
width: '350px',
type: 'Line',
fill: '#007dd1',
dataSource: [3, 6, 4, 1, 3, 2, 5]
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div class="spark" id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Localization in EJ2 JavaScript Sparkline control

The sparkline control supports localization. The default culture for localization is **en-US**. You can change the culture using the **setCulture** method.

Tooltip format

Sparkline tooltip supports localization. The following code sample shows tooltip text with currency format based on culture.

INDEX.TS

```
import { Sparkline, SparklineTooltip } from '@syncfusion/ej2-charts';
Sparkline.Inject(SparklineTooltip);
let sparkline: Sparkline = new Sparkline({
  containerArea: {
    border: { color: '#033e96', width: 2 }
  },
  tooltipSettings: {
    visible: true
  },
  // To specify currency format
  format: 'c0',
  useGroupingSeparator: true,
  lineWidth: 3,
  padding: { left: 20, right: 20, bottom: 20, top: 20 },
  border: { color: '#033e96', width: 2 },
  height: '200px',
  width: '350px',
  type: 'Area',
  fill: '#b2cfff',
  dataSource: [30000, 60000, 40000, 10000, 30000, 20000, 50000]
});
sparkline.appendTo("#element");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div class="spark" id="element" align="center"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

RTL

If you set the `enableRtl` property to true, then the sparkline will be rendered from right-to-left direction.

The following example shows the sparkline is render from "Right-to-left".

INDEX.TS

```

import { Sparkline } from '@syncfusion/ej2-charts';
let sparkline: Sparkline = new Sparkline({
  height: '150px',
  width: '150px',
  enableRtl: true,
  dataSource: [0, 6, 4, 1, 3, 2, 5]
});
sparkline.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 for Sparkline </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Sparkline UI
Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
charts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" style="margin-top: 15%;">
    <div class="spark" id="element" align="center"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Accessibility in EJ2 JavaScript Sparkline control

The Sparkline control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Sparkline control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

WAI-ARIA attributes

The Sparkline control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Sparkline control:

- `img` (role)
- `aria-label` (attribute)
- `aria-hidden` (attribute)

Keyboard interaction

The Sparkline control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Sparkline control.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + P** | Prints the Sparkline. |

Ensuring accessibility

The Sparkline control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Sparkline control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Sparkline control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

Ej1 api migration in EJ2 JavaScript Sparkline control

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Sparkline Types

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Type| **Property:** `type`
`$("#container").ejSparkline({ type : 'line' });` | **Property:** `type`
`let sparkline: Sparkline = new Sparkline({
 type: 'Line'
 });`

`sparkline.appendTo('#container');` |

Databinding

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Datasource| **Property:** `dataSource`
`$("#container").ejSparkline({ dataSource : 'data' });` | **Property:** `dataSource`
`let sparkline: Sparkline = new Sparkline({
 dataSource: data
 });`

`sparkline.appendTo('#container');` |

| Binding X values with datasource| **Property:** *xName*

 `$("#container").ejSparkline({ xName : "XValue" });` | **Property:** *xName*

 `let sparkline: Sparkline = new Sparkline({
 xName: 'XValue'
 });`
 `sparkline.appendTo('#container');` |

| Binding Y values with datasource| **Property:** *yName*

 `$("#container").ejSparkline({ yName : "YValue" });` | **Property:** *yName*

 `let sparkline: Sparkline = new Sparkline({
 yName: 'YValue'
 });`
 `sparkline.appendTo('#container');` |

Markers

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable markers| **Property:** *markerSettings.visible*

 `$("#container").ejSparkline({
 markerSettings: { visible : true }
 });` | **Property:** *markerSettings.visible*

 `let sparkline: Sparkline = new Sparkline({
 markerSettings: {
 visible: ['All']
 }
 });`
 `sparkline.appendTo('#container');` |

| Color| **Property:** *markerSettings.fill*

 `$("#container").ejSparkline({
 markerSettings: { fill : "green" }
 });` | **Property:** *markerSettings.fill*

 `let sparkline: Sparkline = new Sparkline({
 markerSettings: {
 fill: "green"
 }
 });`
 `sparkline.appendTo('#container');` |

| Size| **Property:** *markerSettings.width*

 `$("#container").ejSparkline({
 markerSettings: { width : 5 }
 });` | **Property:** *markerSettings.size*

 `let sparkline: Sparkline = new Sparkline({
 markerSettings: {
 size: 5
 }
 });`
 `sparkline.appendTo('#container');` |

| Opacity| **Property:** *markerSettings.opacity*

 `$("#container").ejSparkline({
 markerSettings: { opacity : 0.5 }
 });` | **Property:** *markerSettings.opacity*

 `let sparkline: Sparkline = new Sparkline({
 markerSettings: {
 opacity: 0.5
 }
 });`
 `sparkline.appendTo('#container');` |

| Border color| **Property:** *markerSettings.border.color*

 `$("#container").ejSparkline({
 markerSettings: {
 border: { color: "green" } }
 });` | **Property:** *markerSettings.border.fill*

 `let sparkline: Sparkline = new Sparkline({
 markerSettings: {
 border: { fill: 'green' }
 }
 });`
 `sparkline.appendTo('#container');` |

| Border width| **Property:** *markerSettings.border.width*

 `$("#container").ejSparkline({
 markerSettings: {
 border: { width: 2 } }
 });` | **Property:** *markerSettings.border.width*

 `let sparkline: Sparkline = new Sparkline({
 markerSettings: {
 border: { width: 2 }
 }
 });`
 `sparkline.appendTo('#container');` |

| Border opacity| **Property:** *markerSettings.border.opacity*

 `$("#container").ejSparkline({
 markerSettings: {
 border: { opacity: 0.7 } }
 });` | Not applicable |

Data labels

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Enable data labels| Not applicable |**Property:** *dataLabelSettings.visible*
 Sparkline = new Sparkline({ dataLabelSettings: { visible: ['All'] } };
 sparkline.appendTo('#container');

|Color| Not applicable |**Property:** *dataLabelSettings.fill*
 Sparkline = new Sparkline({ dataLabelSettings: { fill: "green" } };
 sparkline.appendTo('#container');

|Size| Not applicable |**Property:** *dataLabelSettings.size*
 Sparkline = new Sparkline({ dataLabelSettings: { size: 5 } };
 sparkline.appendTo('#container');

|Opacity| Not applicable |**Property:** *dataLabelSettings.opacity*
 Sparkline = new Sparkline({ dataLabelSettings: { opacity: 0.5 } };
 sparkline.appendTo('#container');

|Border color| Not applicable |**Property:** *dataLabelSettings.border.fill*
 Sparkline = new Sparkline({ dataLabelSettings: { border: { fill: 'green' } } };
 sparkline.appendTo('#container');

|Border width| Not applicable |**Property:** *dataLabelSettings.border.width*
 Sparkline = new Sparkline({ dataLabelSettings: { border: { width: 2 } } };
 sparkline.appendTo('#container');

|Format| Not applicable |**Property:** *dataLabelSettings.format*
 Sparkline = new Sparkline({ dataLabelSettings: { format: '\$ {xval}: \$ {yval}' } };
 sparkline.appendTo('#container');

|Horizontal Offset| Not applicable |**Property:** *dataLabelSettings.offset.x*
 Sparkline = new Sparkline({ dataLabelSettings: { offset: { x: 10 } } };
 sparkline.appendTo('#container');

|Vertical Offset| Not applicable |**Property:** *dataLabelSettings.offset.y*
 Sparkline = new Sparkline({ dataLabelSettings: { offset: { y: 10 } } };
 sparkline.appendTo('#container');

|Font color| Not applicable |**Property:** *dataLabelSettings.textStyle.color*
 Sparkline = new Sparkline({ dataLabelSettings: { textStyle: { color: 'black' } } };
 sparkline.appendTo('#container');

|Font family| Not applicable |**Property:** *dataLabelSettings.textStyle.fontFamily*
 Sparkline = new Sparkline({ dataLabelSettings: { textStyle: { fontFamily: 'Arial' } } };
 sparkline.appendTo('#container');

|Font style| Not applicable |**Property:** *dataLabelSettings.textStyle.fontStyle*
 Sparkline = new Sparkline({ dataLabelSettings: { textStyle: { fontStyle: 'normal' } } };
 sparkline.appendTo('#container');

|Font weight| Not applicable |**Property:** *dataLabelSettings.textStyle.fontWeight*
 Sparkline = new Sparkline({ dataLabelSettings: { textStyle: { fontWeight: 'bold' } } };
 sparkline.appendTo('#container');

|Font opacity| Not applicable |**Property:** *dataLabelSettings.textStyle.opacity*
 Sparkline = new Sparkline({ dataLabelSettings: { textStyle: { opacity: 0.7 } } };
 sparkline.appendTo('#container');

|Font size| Not applicable | **Property:** *dataLabelSettings.textStyle.fontSize*
 Sparkline = new Sparkline({ dataLabelSettings: { textStyle: { fontSize: '12px' } };
 sparkline.appendTo('#container');

Range band

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Color| **Property:** *rangeBandSettings.fill*
 rangeBandSettings :{ fill : "green" }; | **Property:** *rangeBandSettings.fill*
 Sparkline = new Sparkline({ rangeBandSettings: { fill: "green" };
 sparkline.appendTo('#container');

|Opacity| **Property:** *rangeBandSettings.opacity*
 rangeBandSettings :{ opacity : 0.5 }; | **Property:** *rangeBandSettings.opacity*
 sparkline: Sparkline = new Sparkline({ rangeBandSettings: { opacity: 0.5 };
 sparkline.appendTo('#container');

|Start range| **Property:** *rangeBandSettings.startRange*
 rangeBandSettings :{ startRange: 5 }; | **Property:** *rangeBandSettings.startRange*
 let sparkline: Sparkline = new Sparkline({ rangeBandSettings: { startRange: 5 };
 sparkline.appendTo('#container');

|End range| **Property:** *rangeBandSettings.endRange*
 rangeBandSettings :{ endRange: 10 }; | **Property:** *rangeBandSettings.endRange*
 let sparkline: Sparkline = new Sparkline({ rangeBandSettings: { endRange: 10 };
 sparkline.appendTo('#container');

Special points customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|High point color| **Property:** *highPointColor*
 highPointColor: 'green' }; | **Property:** *highPointColor*
 Sparkline = new Sparkline({ highPointColor: 'green' };
 sparkline.appendTo('#container');

|Low point color| **Property:** *lowPointColor*
 lowPointColor: 'red' }; | **Property:** *lowPointColor*
 Sparkline = new Sparkline({ lowPointColor: 'red' };
 sparkline.appendTo('#container');

|Negative point color| **Property:** *negativePointColor*
 negativePointColor: 'orange' }; | **Property:** *negativePointColor*
 Sparkline = new Sparkline({ negativePointColor: 'orange' };
 sparkline.appendTo('#container');

|Start point color| **Property:** *startPointColor*
 startPointColor: 'orange' }; | **Property:** *startPointColor*
 Sparkline = new Sparkline({ startPointColor: 'orange' };
 sparkline.appendTo('#container');

|End point color| **Property:** *endPointColor*
 endPointColor: 'orange' }; | **Property:** *endPointColor*
 Sparkline = new Sparkline({ endPointColor: 'orange' };
 sparkline.appendTo('#container');

| Tie point color| **Property:** *tiePointColor*

 \$(" #container").ejSparkline({ tiePointColor: 'orange' }); | **Property:** *tiePointColor*

 let sparkline: Sparkline = new Sparkline({ tiePointColor: 'orange' });
 sparkline.appendTo('#container');|

Axis customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show axis line| **Property:** *axisSettings.visible*

 \$(" #container").ejSparkline({
 axisSettings: {
 visible: true
 }
 }); | **Property:** *axisSettings.lineSettings.visible*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 lineSettings: { visible: true }
 }
 });
 sparkline.appendTo('#container');|

| Line color| **Property:** *axisSettings.color*

 \$(" #container").ejSparkline({
 axisSettings: {
 color: "green"
 }
 }); | **Property:** *axisSettings.lineSettings.color*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 lineSettings: { color: "green" }
 }
 });
 sparkline.appendTo('#container');|

| Line width| **Property:** *axisSettings.width*

 \$(" #container").ejSparkline({
 axisSettings: {
 width: 2
 }
 }); | **Property:** *axisSettings.lineSettings.width*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 lineSettings: { width: 2 }
 }
 });
 sparkline.appendTo('#container');|

| Dash array| **Property:** *axisSettings.dashArray*

 \$(" #container").ejSparkline({
 axisSettings: {
 dashArray: [5, 2]
 }
 }); | **Property:** *axisSettings.lineSettings.dashArray*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 lineSettings: { dashArray: [5, 2] }
 }
 });
 sparkline.appendTo('#container');|

| X axis minimum value| Not applicable | **Property:** *axisSettings.minX*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 minX: 0
 }
 });
 sparkline.appendTo('#container');|

| X axis maximum value| Not applicable | **Property:** *axisSettings.maxX*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 maxX: 100
 }
 });
 sparkline.appendTo('#container');|

| Y axis minimum value| Not applicable | **Property:** *axisSettings.minY*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 minY: 0
 }
 });
 sparkline.appendTo('#container');|

| Y axis maximum value| Not applicable | **Property:** *axisSettings.maxY*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 maxY: 0
 }
 });
 sparkline.appendTo('#container');|

| Horizontal axis line position| Not applicable | **Property:** *axisSettings.value*

 let sparkline: Sparkline = new Sparkline({
 axisSettings: {
 value: 10
 }
 });
 sparkline.appendTo('#container');|

Appearance customization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Background color | **Property:** *background*
 \$("#container").ejSparkline({ background: "gray" });
Property: *containerArea.background*
 let sparkline: Sparkline = new Sparkline({
 containerArea: { background: "gray" }
 });
 sparkline.appendTo('#container');

| Border color | Not applicable | **Property:** *containerArea.border.color*
 let sparkline: Sparkline = new Sparkline({
 containerArea: {
 border: { color: "green" }
 };
 });
 sparkline.appendTo('#container');

| Border width | Not applicable | **Property:** *containerArea.border.width*
 let sparkline: Sparkline = new Sparkline({
 containerArea: {
 border: { width: 2 }
 };
 });
 sparkline.appendTo('#container');

| Series color | **Property:** *fill*
 \$("#container").ejSparkline({ fill: "green" });
Property: *fill*
 let sparkline: Sparkline = new Sparkline({ fill : "green" });
 sparkline.appendTo('#container');

| Series opacity | **Property:** *opacity*
 \$("#container").ejSparkline({ opacity: 0.5 });
Property: *opacity*
 let sparkline: Sparkline = new Sparkline({ opacity : 0.5 });
 sparkline.appendTo('#container');

| Line series width | **Property:** *width*
 \$("#container").ejSparkline({ width: 3 });
Property: *lineWidth*
 let sparkline: Sparkline = new Sparkline({ lineWidth : 3 });
 sparkline.appendTo('#container');

| Series border color | **Property:** *border.color*
 \$("#container").ejSparkline({ border: { color: "green" } });
Property: *border.color*
 let sparkline: Sparkline = new Sparkline({ border: { color: "green" } });
 sparkline.appendTo('#container');

| Series border width | **Property:** *border.width*
 \$("#container").ejSparkline({ border: { width: 2 } });
Property: *border.width*
 let sparkline: Sparkline = new Sparkline({ border: { width: 2 } });
 sparkline.appendTo('#container');

| Series palette | **Property:** *palette*
 \$("#container").ejSparkline({ palette: "colorFieldNameInDatasource" });
Property: *palette*
 let sparkline: Sparkline = new Sparkline({ palette : ["green", "red", "yellow"] });
 sparkline.appendTo('#container');

| Theme | **Property:** *theme*
 \$("#container").ejSparkline({ theme: "Azure" });
Property: *theme*
 let sparkline: Sparkline = new Sparkline({ theme : "Material" });
 sparkline.appendTo('#container');

| Width | **Property:** *size.width*
 \$("#container").ejSparkline({ size: { width: "100%" } });
Property: *width*
 let sparkline: Sparkline = new Sparkline({ width : "100%" });
 sparkline.appendTo('#container');

| Height | **Property:** *size.height*
 \$("#container").ejSparkline({ size: { height: "200px" } });
Property: *height*
 let sparkline: Sparkline = new Sparkline({ height : "200px" });
 sparkline.appendTo('#container');

Tooltip

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

|Show tooltip| **Property:** *tooltip.visible*

 \$("#container").ejSparkline(({
 tooltip: { visible: true }
 }); | **Property:** *tooltipSettings.visible*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: { visible: true }
 });
 sparkline.appendTo('#container');|

|Background| **Property:** *tooltip.fill*

 \$("#container").ejSparkline(({
 tooltip: { fill: "white" }
 }); | **Property:** *tooltipSettings.fill*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: { fill: "white" }
 });
 sparkline.appendTo('#container');|

|Format| Not applicable | **Property:** *tooltipSettings.format*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: { format: "{\$xval}: {\$yval}" }
 });
 sparkline.appendTo('#container');|

|Template| **Property:** *tooltip.template*

 \$("#container").ejSparkline(({
 tooltip: { template: "tooltip" }
 }); | **Property:** *tooltipSettings.template*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: { template: "tooltip" }
 });
 sparkline.appendTo('#container');|

|Font color| **Property:** *tooltip.font.color*

 \$("#container").ejSparkline(({
 tooltip: {
 font: { color: "gray" }
 }
 }); | **Property:** *tooltipSettings.textStyle.color*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: {
 textStyle: { color: "gray" }
 }
 });
 sparkline.appendTo('#container');|

|Font opacity| **Property:** *tooltip.font.opacity*

 \$("#container").ejSparkline(({
 tooltip: {
 font: { opacity: 0.8 }
 }
 }); | **Property:** *tooltipSettings.textStyle.opacity*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: {
 textStyle: { opacity: 0.8 }
 }
 });
 sparkline.appendTo('#container');|

|Font size| **Property:** *tooltip.font.size*

 \$("#container").ejSparkline(({
 tooltip: {
 font: { size: "14px" }
 }
 }); | **Property:** *tooltipSettings.textStyle.size*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: {
 textStyle: { size: "14px" }
 }
 });
 sparkline.appendTo('#container');|

|Font family| **Property:** *tooltip.font.fontFamily*

 \$("#container").ejSparkline(({
 tooltip: {
 font: { fontFamily: "Arial" }
 }
 }); | **Property:** *tooltipSettings.textStyle.fontFamily*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: {
 textStyle: { fontFamily: "Arial" }
 }
 });
 sparkline.appendTo('#container');|

|Font style| **Property:** *tooltip.font.fontStyle*

 \$("#container").ejSparkline(({
 tooltip: {
 font: { fontStyle: "normal" }
 }
 }); | **Property:** *tooltipSettings.textStyle.fontStyle*

 let sparkline: Sparkline = new Sparkline(({
 tooltipSettings: {
 textStyle: { fontStyle: "normal" }
 }
 });
 sparkline.appendTo('#container');|

|Font weight| **Property:** *tooltip.font.fontWeight*

 \$("#container").ejSparkline(({
 tooltip: {
 font: { fontWeight: "bold" }
 }
 }); | **Property:** *tooltipSettings.textStyle.fontWeight*

 let sparkline: Sparkline = new Sparkline(({


```
tooltipSettings: { <br/> &#160;&#160; textStyle: { fontWeight: "bold" } <br/> &#160; } <br/> }; <br/>
sparkline.appendTo('#container');
```

| Enable track line| Not applicable | **Property:** *tooltipSettings.trackLineSettings.visible*

 let
 sparkline: Sparkline = new Sparkline({
 tooltipSettings: {

 trackLineSettings: { visible: true }
 }
 });
 sparkline.appendTo('#container');

| Track line color| Not applicable | **Property:** *tooltipSettings.trackLineSettings.color*

 let
 sparkline: Sparkline = new Sparkline({
 tooltipSettings: {

 trackLineSettings: { color: "red" }
 }
 });
 sparkline.appendTo('#container');

| Track line width| Not applicable | **Property:** *tooltipSettings.trackLineSettings.width*

 let
 sparkline: Sparkline = new Sparkline({
 tooltipSettings: {

 trackLineSettings: { width: 2 }
 }
 });
 sparkline.appendTo('#container');

Rendering

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable canvas rendering| **Property:** *enableCanvasRendering*

 `$("#container").ejSparkline({
 enableCanvasRendering : true });` | Not applicable |

Localization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Localization| **Property:** *locale*

 `$("#container").ejSparkline({ locale : "en-US" });` | **Property:**
type

 let sparkline: Sparkline = new Sparkline({ locale: 'en-US' });

 sparkline.appendTo('#container');

Methods

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Dynamically updating sparkline| **Method:** *redraw*

 `$("#container").ejSparkline("redraw");` |
Method: *refresh*

 `sparkline.refresh();` |

| Append sparkline to element| Not applicable | **Method:** *appendTo*

`sparkline.appendTo("#container");` |

Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Load| **Event:** *load*

 `$("#container").ejSparkline({
 load: function(args) { }
 });`
 | **Event:** *load*

 let sparkline: Sparkline = new Sparkline({
 load: function(e:
 ISparklineLoadEventArgs): void { }
});
 sparkline.appendTo('#container');

| Load completed| **Event:** *loaded*

 `$("#container").ejSparkline({
 loaded:
 function(args) { }
 });` | **Event:** *loaded*

 let sparkline: Sparkline = new Sparkline({

 loaded: function(e: ISparklineLoadedEventArgs): void { }
});

 sparkline.appendTo('#container');

|Initialize tooltip| **Event:** *tooltipInitialize*

 \$("#container").ejSparkline({
 tooltipInitialize: function(args) { }
 }); | **Event:** *tooltipInitialize*

 let sparkline: Sparkline = new Sparkline({
 tooltipInitialize: function(e: ITooltipRenderingEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|Series rendering| **Event:** *seriesRendering*

 \$("#container").ejSparkline({
 seriesRendering: function(args) { }
 }); | **Event:** *seriesRendering*

 let sparkline: Sparkline = new Sparkline({
 seriesRendering: function(e: ISeriesRenderingEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|Region mouse move| **Event:** *pointRegionMouseMove*

 \$("#container").ejSparkline({
 pointRegionMouseMove: function(args) { }
 }); | **Event:** *pointRegionMouseMove*

 let sparkline: Sparkline = new Sparkline({
 pointRegionMouseMove: function(e: IPointRegionEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|Region click| **Event:** *pointRegionMouseClick*

 \$("#container").ejSparkline({
 pointRegionMouseClick: function(args) { }
 }); | **Event:** *pointRegionMouseClick*

 let sparkline: Sparkline = new Sparkline({
 pointRegionMouseClick: function(e: IPointRegionEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|Mouse move| **Event:** *sparklineMouseMove*

 \$("#container").ejSparkline({
 sparklineMouseMove: function(args) { }
 }); | **Event:** *sparklineMouseMove*

 let sparkline: Sparkline = new Sparkline({
 sparklineMouseMove: function(e: ISparklineMouseEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|Mouse leave| **Event:** *sparklineMouseLeave*

 \$("#container").ejSparkline({
 sparklineMouseLeave: function(args) { }
 }); | Not applicable |

|Click| **Event:** *click*

 \$("#container").ejSparkline({
 click: function(args) { }
 }); | **Event:** *sparklineMouseClick*

 let sparkline: Sparkline = new Sparkline({
 sparklineMouseClick: function(e: ISparklineMouseEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|doubleClick| **Event:** *doubleClick*

 \$("#container").ejSparkline({
 doubleClick: function(args) { }
 }); | Not applicable |

|rightClick| **Event:** *rightClick*

 \$("#container").ejSparkline({
 rightClick: function(args) { }
 }); | Not applicable |

|axisRendering| Not applicable | **Event:** *axisRendering*

 let sparkline: Sparkline = new Sparkline({
 axisRendering: function(e: IAxisRenderingEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|dataLabelRendering| Not applicable | **Event:** *dataLabelRendering*

 let sparkline: Sparkline = new Sparkline({
 dataLabelRendering: function(e: IDataLabelRenderingEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|markerRendering| Not applicable | **Event:** *markerRendering*

 let sparkline: Sparkline = new Sparkline({
 markerRendering: function(e: IMarkerRenderingEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|pointRendering| Not applicable | **Event:** *pointRendering*
 Sparkline({
pointRendering: function(e: ISparklinePointEventArgs): void { }
});
 sparkline.appendTo('#container'); |

|resize| Not applicable | **Event:** *resize*
 Sparkline({
resize: function(e: ISparklineResizeEventArgs): void { }
});
 sparkline.appendTo('#container'); |

SpeedDial

Items in EJ2 JavaScript Speed dial control

The JavaScript(ES5) Speed Dial action items can be added by using [items](#) property.

| Fields | Type | Description |

|-----|-----|-----|

| [text](#) | string | Defines the text content of SpeedDialItem. |

| [iconCss](#) | string | Defines one or more CSS classes to include an icon or image in Speed Dial item. |

| [disabled](#) | boolean | Defines whether to enable or disable the SpeedDialItem. |

| [id](#) | string | Defines a unique value for the SpeedDialItem which can be used to identify the item in event args. |

| [title](#) | string | Defines the title of SpeedDialItem to display tooltip. |

Icons in SpeedDial items

You can customize the icon and text of Speed Dial action items using [iconCss](#) and [text](#) properties.

Icon only

You can show icon only in SpeedDial items by setting [iconCss](#) property. You can show tooltip on hover to show additional details to end-user by setting [title](#) property.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items with icon only
var items = [
  { iconCss: 'e-icons e-cut', title: 'Cut' },
  { iconCss: 'e-icons e-copy', title: 'Copy' },
  { iconCss: 'e-icons e-paste', title: 'Paste' }
];
// Initialize the SpeedDial control
var speedDial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  closeIconCss: 'e-icons e-close',
  target: '#targetElement'
});
// Render initialized SpeedDial
speedDial.appendTo('#speeddial');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <button id="speeddial"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Text only

You can show only text in Speed Dial items by setting [text](#) property.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize action items with text only
var items = [
    { text: 'Cut' },
    { text: 'Copy' },

```

```

    { text: 'Paste' }
  ];
  // Initialize the SpeedDial control
  var speedDial = new ej.buttons.SpeedDial({
    items: items,
    content: 'Edit',
    target: '#targetElement'
  });
  // Render initialized SpeedDial
  speedDial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;

```

```

position: absolute;
top: 45%;
width: 30%;
}

```

Icon with text

You can show icon along with text in Speed Dial items by setting [iconCss](#) and [text](#) properties.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize action items with icon and text
var items = [
    { text: 'Cut', iconCss: 'e-icons e-cut' },
    { text: 'Copy', iconCss: 'e-icons e-copy' },
    { text: 'Paste', iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial control
var speedDial = new ej.buttons.SpeedDial({
    items: items,
    openIconCss: 'e-icons e-edit',
    closeIconCss: 'e-icons e-close',
    content: 'Edit',
    target: '#targetElement'
});
// Render initialized SpeedDial
speedDial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
</script>

```

```
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Disabled

You can disable Speed Dial items by setting [disabled](#) property as `true`.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items with disabled first item.
var items = [
    { text: 'Cut', disabled: true },
    { text: 'Copy' },
    { text: 'Paste' }
];
// Initialize the SpeedDial control
var speedDial = new ej.buttons.SpeedDial({
    items: items,
    content: 'Edit',
    target: '#targetElement'
});
// Render initialized SpeedDial
speedDial.appendTo('#speeddial');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SplitButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <button id="speeddial"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Animation

The Speed Dial items can be animated during the opening and closing of the popup action items. You can customize the animation's effect, delay, and duration by setting [animation](#) property. By default, Speed Dial animates with a fade effect and supports all [speeddialanimation](#) effects.

Below example demonstrates the Speed Dial items with applied Zoom effect.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize action items
var items = [
    { text: 'Cut', iconCss: 'e-icons e-cut' },
    { text: 'Copy', iconCss: 'e-icons e-copy' },
    { text: 'Paste', iconCss: 'e-icons e-paste' }
];
// Intialize animation effect

```

```

var animation = { effect: 'Zoom' };
// Initialize the SpeedDial component
var speedDial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  closeIconCss: 'e-icons e-close',
  content: 'Edit',
  animation: animation,
  target: '#targetElement'
});
// Render initialized SpeedDial
speedDial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
}

```

```
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
```

Template

The Speed Dial supports to customize the action items and entire pop-up container by setting [itemTemplate](#) and [popupTemplate](#) properties. For more details about templates, check out the link [here](#).

Positions in EJ2 JavaScript Speed dial control

The Speed dial control can be positioned anywhere on the [target](#) using the [position](#) property. If the [target](#) is not defined, then Speed Dial is positioned based on the browser viewport.

The position values of Speed Dial are as follows:

- TopLeft
- TopCenter
- TopRight
- MiddleLeft
- MiddleCenter
- MiddleRight
- BottomLeft
- BottomCenter
- BottomRight

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items
var items = [
  { iconCss: 'e-icons e-cut' },
  { iconCss: 'e-icons e-copy' },
  { iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial component in linear mode
var linearSpeeddial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  mode: 'Linear',
  direction: 'Left',
  target: '#targetElement'
});
// Render initialized SpeedDial
linearSpeeddial.appendTo('#speeddial');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<!-- Add the SpeedDial component styles. -->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <button id="speeddial" title="Edit"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Opens items on hover

You can open the Speed Dial action items on mouse hover by setting the [opensOnHover](#) property.

```

`js
// Initialize action items
var items = [

```

```
{ iconCss: 'e-icons e-cut' },
{ iconCss: 'e-icons e-copy' },
{ iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial control
var speedDial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  closeIconCss: 'e-icons e-close',
  opensOnHover: true,
  target: '#targetElement'
});
// Render initialized SpeedDial
speedDial.appendTo('#speedDial');
```

Programmatically show/hide SpeedDial items

You can open/close the Speed Dial action items programmatically using [show](#) and [hide](#) methods.

Below example demonstrates open/close action items on button click.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items
var items = [
  { iconCss: 'e-icons e-cut' },
  { iconCss: 'e-icons e-copy' },
  { iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial component
var speedDial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  closeIconCss: 'e-icons e-close',
  opensOnHover: true,
  target: '#targetElement'
});
// Render initialized SpeedDial
speedDial.appendTo('#speeddial');
// Show and Hide button click event handler
document.getElementById('show').addEventListener('click', function () {
  speedDial.show();
});
document.getElementById('hide').addEventListener('click', function() {
  speedDial.hide();
});
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- Elements to render the SpeedDial component. -->
    <div id="targetElement" style="position:relative;min-
height:340px;border:1px solid;padding:10px;">
      <button id="show">Show</button>
      <button id="hide" style="float:right">Hide</button></div>
      <button id="speeddial"></button>
    </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
button{
  background-color: #aaa;

```

```
border: none;
border-radius: 5px;
color: black;
padding: 13px 28px;
text-align: center;
display: inline-block;
font-size: 15px;
}
```

Programmatically refresh the position

You can refresh the position of the Speed Dial using [refreshPosition](#) method when the **target** position is changed.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items
var items = [
  { iconCss: 'e-icons e-cut' },
  { iconCss: 'e-icons e-copy' },
  { iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial control
var speedDial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  closeIconCss: 'e-icons e-close',
  position: 'MiddleRight',
  target: '#targetElement'
});
// Render initialized SpeedDial
speedDial.appendTo('#speeddial');
// Refresh button click event handler
document.getElementById('refresh').addEventListener('click', function () {
  document.getElementById("targetElement").style.minHeight = "300px";
  speedDial.refreshPosition();
});
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <div id="targetElement" style="position:relative;min-
height:340px;border:1px solid;padding:10px;">
            <button id="refresh">Refresh</button></div>
            <button id="speeddial"></button>
        </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
button{
    background-color: #aaa;
    border: none;
    border-radius: 5px;
    color: black;
    padding: 13px 28px;
    text-align: center;
    display: inline-block;
    font-size: 15px;
}

```

Display modes in EJ2 JavaScript Speed dial control

The action items in JavaScript(ES5) Speed Dial can be displayed in **Linear** and **Radial** display modes by setting [mode](#) property.

Linear display mode

In **Linear** display mode, Speed Dial action items are displayed in a list-like format either horizontally or vertically. By default, Speed Dial items are displayed in **Linear** mode.

Direction

You can open the action items on the top, left, up, and down side of the Speed Dial button by setting [direction](#) property. The default value is `Auto` where the action items are displayed based on the [position](#) of the Speed Dial.

The `Linear` directions of Speed Dial are as follows:

- Left - Action items are displayed on the left side of the button.
- Right - Action items are displayed on the right side of the button.
- Up - Action items are displayed on the top of the button.
- Down - Action items are displayed on the bottom of the button.
- Auto - Action items display direction auto calculated based on `position` of the Speed Dial. If Speed Dial is position at bottom right, then action items displayed at top.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items
var items = [
  { iconCss: 'e-icons e-cut' },
  { iconCss: 'e-icons e-copy' },
  { iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial component in linear mode
var linearSpeeddial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  mode: 'Linear',
  direction: 'Left',
  target: '#targetElement'
});
// Render initialized SpeedDial
linearSpeeddial.appendTo('#speeddial');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
```

```

</head>
<body>

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <button id="speeddial" title="Edit"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Radial display mode (Radial Menu)

In **Radial** mode, Speed Dial action items are displayed in a circular pattern like a radial menu. For more details about radial mode, check out the link [here](#).

Radial menu in EJ2 JavaScript Speed dial control

The JavaScript(ES5) Speed Dial action items can be displayed in a circular pattern like a radial menu by setting **mode** property. You can customize the **direction**, **startAngle**, **endAngle** and **offset** by setting **radialSettings** property.

Radial menu direction

You can open the action items in either clockwise or anticlockwise by setting **direction** property. The default value is **Auto** where the action items are displayed based on the **position** property of the Speed Dial.

```

`js
// Initialize action items
var items = [
    { iconCss: 'e-icons e-cut' },
    { iconCss: 'e-icons e-copy' },

```

```
{ iconCss: 'e-icons e-paste' }  
];  
// Initialize the SpeedDial control in anticlockwise direction  
var radialSettings = { direction: 'AntiClockwise' };  
// Initialize the SpeedDial control  
var speedDial = new ej.buttons.SpeedDial({  
  items: items,  
  openIconCss: 'e-icons e-edit',  
  target: '#targetElement',  
  mode: 'Radial',  
  radialSettings: radialSettings  
});  
// Render initialized SpeedDial  
speedDial.appendTo('#speedDial');  
`
```

Radial menu start and end angle

You can modify the start and end angle of action items by setting [startAngle](#) and [endAngle](#) properties. If the angle is not defined, the action items are displayed based on the [position](#) property of the Speed Dial.

```
`js  
// Initialize action items  
var items = [  
  { iconCss: 'e-icons e-cut' },  
  { iconCss: 'e-icons e-copy' },  
  { iconCss: 'e-icons e-paste' },  
];  
// Initialize radial direction with start and end angles  
var radialSettings = { direction: 'AntiClockwise', startAngle: 180, endAngle: 360 };  
// Initialize the SpeedDial control  
var speedDial = new ej.buttons.SpeedDial({  
  items: items,  
  openIconCss: 'e-icons e-edit',  
  target: '#targetElement',  
  mode: 'Radial',
```



```
position: 'MiddleCenter',
radialSettings: radialSettings
});
// Render initialized SpeedDial
speedDial.appendTo('#speedDial');
`
```

Offset

You can modify the offset distance between action items and Speed Dial button using [offset](#) property.

```
`js
// Initialize action items
var items = [
{ iconCss: 'e-icons e-cut' },
{ iconCss: 'e-icons e-copy' },
{ iconCss: 'e-icons e-paste' }
];
// Initialize radial direction with offset
var radialSettings = { offset: '80px' };
// Initialize the SpeedDial control
var speedDial = new ej.buttons.SpeedDial({
items: items,
openIconCss: 'e-icons e-edit',
target: '#targetElement',
mode: 'Radial',
radialSettings: radialSettings
});
// Render initialized SpeedDial
speedDial.appendTo('#speedDial');
`
```

Below example demonstrates the radial menu settings of the Speed Dial.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items
var items = [
  { iconCss: 'e-icons e-cut' },
  { iconCss: 'e-icons e-copy' },
  { iconCss: 'e-icons e-paste' },
```

```

    { iconCss:'e-icons e-edit'},
    { iconCss:'e-icons e-save'}
  ];
  // Initialize radial direction
  var radialSettings = { offset: '80px', direction: 'AntiClockwise',
    startAngle: 90, endAngle: 270 };
  // Initialize the SpeedDial component
  var speedDial = new ej.buttons.SpeedDial({
    items: items,
    openIconCss:'e-icons e-edit',
    target: '#targetElement',
    mode: 'Radial',
    position: 'MiddleRight',
    radialSettings: radialSettings
  });
  // Render initialized SpeedDial
  speedDial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- Elements to render the SpeedDial component. -->
    <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Template in EJ2 JavaScript Speed dial control

This section explains available templates in SpeedDial control and its usage.

Item template

You can use the [itemTemplate](#) property to set a template content for the SpeedDial items. The template content is defined as a child content of `itemTemplate` tag directive.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize action items
var items = [
    { text: 'Cut', iconCss: 'e-icons e-cut' },
    { text: 'Copy', iconCss: 'e-icons e-copy' },
    { text: 'Paste', iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial component
var speeddial = new ej.buttons.SpeedDial({
    items: items,
    openIconCss: 'e-icons e-edit',
    content: 'Edit',
    target: '#targetElement',
    itemTemplate: '#itemTemplate'
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <button id="speeddial" title="Edit"></button>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    </div>
    <script type="text/x-jsrender" id="itemTemplate">
        <div class="itemlist">
            <span class="icon ${iconCss}" style="padding:3px"></span>
            <span class="text">${text}</span>
        </div>
    </script>
    <script type="text/x-jsrender" id="popupTemplate">
        <div>
            <div class="speeddial-form">
                <p>Here you can customize your code.</p>
            </div>
        </div>
    </script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.itemlist .text {
    padding: 0 5px;
}
.e-speeddial-li .itemlist {
    display: inherit;
    width: 100%;
}

```

```

border: 1px solid transparent;
align-items: center;
padding: 5px;
border-radius: 500px;
background-color: rgba(104, 99, 104, 0.1);
box-shadow: 0 0 4px grey;
}
.e-speeddial-li .itemlist:hover {
background-color: rgb(224, 224, 224);
}
.speeddial-form {
width: 200px;
height: 80px;
text-align: center;
border-radius: 15px;
box-shadow: rgb(0 0 0 / 10%) 0px 10px 15px -3px, rgb(0 0 0 / 5%) 0px 4px
6px -2px;
background: #f5f5f5;
padding: 15px;
}

```

Popup template

You can use the [popupTemplate](#) property to set a template content for popup of SpeedDial control. The template content is defined as a child content of `popupTemplate` tag directive.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize action items
var items = [
  { text: 'Cut', iconCss: 'e-icons e-cut' },
  { text: 'Copy', iconCss: 'e-icons e-copy' },
  { text: 'Paste', iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial component
var speeddial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  content: 'Edit',
  target: '#targetElement',
  popupTemplate: '#popupTemplate'
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <button id="speeddial" title="Edit"></button>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    </div>
    <script type="text/x-jsrender" id="itemTemplate">
        <div class="itemlist">
            <span class="icon ${iconCss}"style="padding:3px"></span>
            <span class="text">${text}</span>
        </div>
    </script>
    <script type="text/x-jsrender" id="popupTemplate">
        <div>
            <div class="speeddial-form">
                <p>Here you can customize your code.</p>
            </div>
        </div>
    </script>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.itemlist .text {

```

```
padding: 0 5px;
}
.e-speeddial-li .itemlist {
  display: inherit;
  width: 100%;
  border: 1px solid transparent;
  align-items: center;
  padding: 5px;
  border-radius: 500px;
  background-color: rgba(104, 99, 104, 0.1);
  box-shadow: 0 0 4px grey;
}
.e-speeddial-li .itemlist:hover {
  background-color: rgb(224, 224, 224);
}
.speeddial-form {
  width: 200px;
  height: 80px;
  text-align: center;
  border-radius: 15px;
  box-shadow: rgb(0 0 0 / 10%) 0px 10px 15px -3px, rgb(0 0 0 / 5%) 0px 4px
6px -2px;
  background: #f5f5f5;
  padding: 15px;
}
```

Styles in EJ2 JavaScript Speed dial control

This section briefs different ways to style SpeedDial control.

SpeedDial button

You can customize the icon and text of JavaScript(ES5) Speed Dial button using [openIconCss](#), [closeIconCss](#) and [content](#) properties.

Icon only

You can use the [openIconCss](#) and [closeIconCss](#) properties to show icons in speed dial button. You can also show tooltip on hover to show additional details to end-user by setting `title` attribute.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the SpeedDial control with icon only
var speeddial = new ej.buttons.SpeedDial({
  openIconCss: 'e-icons e-edit',
  closeIconCss: 'e- icons e-close',
  target: '#targetElement'
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<!-- Add the SpeedDial component styles. -->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <button id="speeddial"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Text only

You can show only text in Speed Dial button by setting [content](#) property without setting icon properties..

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the SpeedDial control with text only
var speeddial = new ej.buttons.SpeedDial({

```



```

        content: 'Edit',
        target: '#targetElement'
    });
    // Render initialized SpeedDial.
    speeddial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- Elements to render the SpeedDial component. -->
    <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;

```

```
}

```

Icon with text

You show icon and text in SpeedDial button using [openIconCss](#), [closeIconCss](#) and [content](#) properties together.

INDEX.JS

```
ej.base.enableRipple(true);
// Initialize the SpeedDial control with icon and text
var speeddial = new ej.buttons.SpeedDial({
  content: 'Edit',
  openIconCss: 'e-icons e-edit',
  closeIconCss: 'e- icons e-close',
  target: '#targetElement'
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- Elements to render the SpeedDial component. -->
    <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Disabled

You can enable or disable the SpeedDial control using [disabled](#) property.

```
`js
```

```
// Initialize the SpeedDial control in disabled state
```

```
var speeddial = new ej.buttons.SpeedDial({
```

```
content: 'Edit',
```

```
target: '#targetElement',
```

```
disabled: true
```

```
});
```

```
// Render initialized SpeedDial
```

```
speeddial.appendTo('#speeddial');
```

```
,
```

cssClass

The JavaScript(ES5) Speed Dial supports the following predefined styles that can be defined using the [cssClass](#) property. You can customize by setting the `cssClass` property with the below defined class.

cssClass	Description
-----	-----
e-primary	Used to represent a primary action.
e-outline	Used to represent an appearance of button with outline.
e-info	Used to represent an informative action.
e-success	Used to represent a positive action.
e-warning	Used to represent an action with caution.
e-danger	Used to represent a negative action.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the SpeedDial with applied warning style
var speeddial = new ej.buttons.SpeedDial({
  content: 'Edit',
  target: '#targetElement',
  cssClass: 'e-warning'
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- Elements to render the SpeedDial component. -->
    <div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
}

```

```

left: 45%;
position: absolute;
top: 45%;
width: 30%;
}

```

Visible

You can set the Speed Dial button to visible/hidden state using [visible](#) property.

```

`js
// Initialize the SpeedDial control in hidden state
var speeddial = new ej.buttons.SpeedDial({
  content: 'Edit',
  target: '#targetElement',
  visible: false
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
`

```

Tooltip

You can show tooltip on hover to show additional details to end-user by setting [title](#) to Speed Dial button.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize action items with tooltip
var items = [
  { iconCss: 'e-icons e-cut', title: 'Cut' },
  { iconCss: 'e-icons e-copy', title: 'Copy' },
  { iconCss: 'e-icons e-paste', title: 'Paste' }
];
// Initialize the SpeedDial control
var speeddial = new ej.buttons.SpeedDial({
  openIconCss: 'e-icons e-edit',
  items: items,
  target: '#targetElement'
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<!-- Add the SpeedDial component styles. -->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <button id="speeddial" title="Edit"></button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Opens on hover

You can use [opensOnHover](#) property to open actions items on hover itself. By default action items displayed only when clicking the speed dial button.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize action items
var items = [
    { iconCss: 'e-icons e-cut' },

```

```

    { iconCss: 'e-icons e-copy' },
    { iconCss: 'e-icons e-paste' }
  ];
  // Initialize the SpeedDial control
  let speedDial = new ej.buttons.SpeedDial({
    items: items,
    openIconCss: 'e-icons e-edit',
    closeIconCss: 'e-icons e-close',
    opensOnHover: true,
    target: '#targetElement'
  });
  // Render initialized SpeedDial
  speedDial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!-- Elements to render the SpeedDial component. -->
    <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="speeddial"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}

```

```

}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}

```

Modal in EJ2 JavaScript Speed dial control

You can use the [modal](#) property to set the Speed Dial as modal which adds an overlay to prevent the background interaction.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize action items
var items = [
  { iconCss: 'e-icons e-cut' },
  { iconCss: 'e-icons e-copy' },
  { iconCss: 'e-icons e-paste' }
];
// Initialize the SpeedDial control with modal
var speeddial = new ej.buttons.SpeedDial({
  items: items,
  openIconCss: 'e-icons e-edit',
  target: '#targetElement',
  modal: true
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpeedDial</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <!-- Add the SpeedDial component styles. -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

```



```

    <div id="container">
        <!-- Elements to render the SpeedDial component. -->
        <button id="speeddial" title="Edit"></button>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Event in EJ2 JavaScript Speed dial control

This section explains the Speed Dial events that will be triggered when appropriate actions are performed.

clicked

The SpeedDial control triggers the [clicked](#) event with [SpeedDialItemEventArgs](#) argument when an action item is clicked. You can use this event to perform the required action.

```

`js
// Initialize action items
var items = [
    { text: 'Cut'},
    { text: 'Copy'},
    { text: 'Paste'}
];
// Initialize the SpeedDial control
var speeddial= new ej.buttons.SpeedDial({
    items: items,

```

```
content: 'Edit',
target: '#targetElement',
clicked: (args)=> {
//Your required action here
}
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
```

[created](#)

The Speed Dial control triggers the [created](#) event when SpeedDial control rendering is completed.

```
`js
// Initialize action items
var items = [
{ text: 'Cut'},
{ text: 'Copy'},
{ text: 'Paste'}
];
// Initialize the SpeedDial control
var speeddial: = new ej.buttons.SpeedDial({
items: items,
content: 'Edit',
target: '#targetElement',
created: ()=> {
//Your required action here
}
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
```

[beforeOpen](#)

The SpeedDial control triggers the [beforeOpen](#) event with [SpeedDialBeforeOpenCloseEventArgs](#) argument before the SpeedDial popup is opened.

```
`js
```

```
// Initialize action items
var items = [
  { text: 'Cut'},
  { text: 'Copy'},
  { text: 'Paste'}
];
// Initialize the SpeedDial control
var speeddial = new ej.buttons.SpeedDial({
  items: items,
  content: 'Edit',
  target: '#targetElement',
  beforeOpen: (args)=> {
    //Your required action here
  }
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
```

onOpen

The SpeedDial control triggers the [onOpen](#) event with [SpeedDialOpenCloseEventArgs](#) argument when SpeedDial popup is opened.

```
`js
// Initialize action items
var items = [
  { text: 'Cut'},
  { text: 'Copy'},
  { text: 'Paste'}
];
// Initialize the SpeedDial control
var speeddial = new ej.buttons.SpeedDial({
  items: items,
  content: 'Edit',
  target: '#targetElement',
  onOpen: (args)=> {
```

```
//Your required action here
}
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
`
```

beforeClose

The SpeedDial control triggers the [beforeClose](#) event with [SpeedDialBeforeOpenCloseEventArgs](#) argument before the SpeedDial popup is closed.

```
`js
// Initialize action items
var items = [
{ text: 'Cut'},
{ text: 'Copy'},
{ text: 'Paste'}
];
// Initialize the SpeedDial control
var speeddial = new ej.buttons.SpeedDial({
items: items,
content: 'Edit',
target: '#targetElement',
beforeClose: (args)=> {
//Your required action here
}
});
// Render initialized SpeedDial
speeddial.appendTo('#speeddial');
`
```

onClose

The SpeedDial control triggers the [onClose](#) event with [SpeedDialOpenCloseEventArgs](#) argument when SpeedDial popup is closed.

```
`js
// Initialize action items
var items = [
```

```
{ text: 'Cut'},
{ text: 'Copy'},
{ text: 'Paste'}
];
// Initialize the SpeedDial control
var speeddial = new ej.buttons.SpeedDial({
  items: items,
  content: 'Edit',
  target: '#targetElement',
  onClose: (args)=> {
    //Your required action here
  }
});
// Render initialized SpeedDial.
speeddial.appendTo('#speeddial');
```

[beforeItemRender](#)

The SpeedDial control triggers the [beforeItemRender](#) event with [SpeedDialItemEventArgs](#) argument for each SpeedDialItem once it is rendered.

```
`js
// Initialize action items
var items = [
  { text: 'Cut'},
  { text: 'Copy'},
  { text: 'Paste'}
];
// Initialize the SpeedDial control
var speeddial = new ej.buttons.SpeedDial({
  items: items,
  content: 'Edit',
  target: '#targetElement',
  beforeItemRender: (args)=> {
    //Your required action here
  }
});
```

```
});  
// Render initialized SpeedDial  
speeddial.appendTo('#speeddial');  
`
```

Below example demonstrates the clicked event of the Speed Dial control.

INDEX.JS

```
ej.base.enableRipple(true);  
// Initialize action items  
var items = [  
  { text: 'Cut' },  
  { text: 'Copy' },  
  { text: 'Paste' }  
];  
// Initialize SpeedDial control with clicked event  
var speeddial = new ej.buttons.SpeedDial({  
  items: items,  
  content: 'Edit',  
  target: '#targetElement',  
  clicked: actionClicked  
});  
// Render initialized SpeedDial.  
speeddial.appendTo('#speeddial');  
function actionClicked(args) {  
  alert(args.item.text + " is clicked");  
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
  <title>EJ2 SpeedDial</title>  
  <meta charset="utf-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta name="description" content="Typescript UI Controls">  
  <meta name="author" content="Syncfusion">  
  <!-- Add the SpeedDial component styles. -->  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">  
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">  
  <link href="styles.css" rel="stylesheet">  
  
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>  
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>  
</head>  
<body>  
  
  <div id="container">  
    <!-- Elements to render the SpeedDial component. -->
```

```

        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
        <button id="speeddial" title="Edit"></button>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Accessibility in EJ2 JavaScript Speed dial control

Accessibility is achieved in the Speed Dial control through WAI-ARIA standard and keyboard navigations. The Speed Dial control can be effectively accessed through assistive technologies such as screen readers.

Keyboard interaction

The Speed Dial control is interactive with below keyboard shortcuts.

| Keyboard shortcuts | Actions |

|-----|-----|

| Enter | Open/close the menu. If a SpeedDial item is focused, should triggers the clicked event for the item. |

| Up-arrow | Navigates up or to the previous menu item. |

| Left-Arrow | Navigates left or to the previous menu item. |

| Down-Arrow | Navigates down or to the previous menu item. |

| Right-Arrow | Navigates right or to the previous menu item. |

| Home | Navigates to the first menu item. |

| End | Navigates to the last menu item. |

| Esc | Closes the menu. |

ARIA attributes

The following ARIA attributes are used in SpeedDial control based on its state.

Properties	Functionality
role	This attribute is added to the input element to describe the actual role.
aria-label	Attribute provides the text label with some default description for the SpeedDial and its items.
aria-expanded	It indicates whether the SpeedDial current state is expanded or collapsed.
aria-haspopup	It indicates whether the SpeedDial has popup items or not.
aria-controls	Attribute is set to the SpeedDial button and it points to the corresponding content.
aria-disabled	It indicates the disabled state of the SpeedDial and its items.

Spinner

Template in EJ2 JavaScript Spinner control

You can use custom templates on the Spinner instead of the default Spinner by specifying the template in the `setSpinner` method.

The following steps explain you on how to define template for Spinner.

- Import the `setSpinner` method from `ej2-popups` library into your `app.ts` as shown in below.

```
`ts
import { setSpinner } from '@syncfusion/ej2-popups';
`
```

- Pass your custom template to the `setSpinner` method like as below.

```
`ts
// Specify the template content to be displayed in the Spinner
setSpinner({ template: '<div style="width:100%;height:100%" class="custom-rolling"><div></div></div>'});
`
```

You should set the template to the Spinner before creating the respective Essential JS 2 component. Also, until we replace `setSpinner` template, the further Essential JS 2 component rendering is created with given template only.

- Now, render the Essential JS 2 component. It's render the Spinner with the template specified in the `setSpinner` method.

In the below sample, we have rendered the Grid component with custom Spinner using `setSpinner` method. You have to define the styles for the template in `styles.css`.

INDEX.TS

```
import { createSpinner, setSpinner, showSpinner } from '@syncfusion/ej2-
popups';
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let gridData: Object[] = data.slice(0, 7);
// By default, Spinner is rendered with specified theme inside the Grid
component.
let grid: Grid = new Grid({
    dataSource: gridData,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
'right', width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer
ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
'right', width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140,
format: 'yMd' }
    ]
});
grid.appendTo('#Grid');
grid.hideSpinner = () => true;
// Specify the template content to be displayed in the Spinner
setSpinner({ template: '<div style="width:100%;height:100%" class="custom-
rolling"><div></div></div>' });
let gridData2: Object[] = data.slice(0, 7);
// Spinner is rendered with template specified in the setSpinner method.
let grid2: Grid = new Grid({
    dataSource: gridData2,
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign: 'right',
width: 120, type: 'number' },
        { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
        { field: 'Freight', headerText: 'Freight', textAlign: 'right',
width: 120, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
    ]
});
grid2.appendTo('#Grid2');
grid2.hideSpinner = () => true;
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Spinner</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="TypeScript NumericTextBox Component">
    <meta name="author" content="Syncfusion">
```

```

<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <h5> Spinner is rendered with specified theme</h5>
        <div id="Grid" class="spinner-grid">
        </div>
        <h5> Spinner is rendered with specified template in setSpinner</h5>
        <div id="Grid2" class="spinner-grid">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Types in EJ2 JavaScript Spinner control

By default, the Spinner is loaded in the applicable Essential JS 2 component based on the theme imported into the page. Based on the theme, the type is set to the Spinner.

The available types are:

- Material
- Fabric
- Bootstrap

You can change the Essential JS 2 component spinner type by passing the type of the spinner as parameter to the `setSpinner` method like as below.

```

`ts
// Specify the type of the Spinner to be displayed
setSpinner({ type: 'Bootstrap'});
`

```

After Essential JS 2 component creation only, you can change the Essential JS 2 component spinner type.

INDEX.TS

```
import { createSpinner, setSpinner , showSpinner } from '@syncfusion/ej2-
popups';
import { Grid } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
let gridData: Object[] = data.slice(0, 7);
// By default, Spinner is rendered with specified theme inside the Grid
component.
let grid: Grid = new Grid({
  dataSource: gridData,
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign: 'right',
width: 120, type: 'number' },
    { field: 'CustomerID', width: 140, headerText: 'Customer ID', type:
'string' },
    { field: 'Freight', headerText: 'Freight', textAlign: 'right',
width: 120, format: 'C' },
    { field: 'OrderDate', headerText: 'Order Date', width: 140, format:
'yMd' }
  ]
});
grid.appendTo('#Grid');
grid.hideSpinner = () => true;
// Specify the type of the Spinner to be displayed in the Grid.
setSpinner({ type: 'Bootstrap' });
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Spinner</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="TypeScript NumericTextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <h5> Grid is rendered with Bootstrap type Spinner </h5>
```

```

        <div id="Grid" class="spinner-grid">
        </div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Style in EJ2 JavaScript Spinner control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the spinner

Use the following CSS to customize the spinner stroke color.

Material theme

```

`css
.e-spinner-pane .e-spinner-inner .e-spin-material {
stroke: green;
}
`

```

Fabric theme

```

`css
.e-spinner-pane .e-spinner-inner .e-spin-fabric {
stroke: green;
}
`

```

Bootstrap theme

```

`css
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap {
fill: green;
stroke: green;
}
`

```

Bootstrap4 theme

```

`css
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap4 {
stroke: green;
}
`

```

```

}
`

```

High Contrast theme

```
`css
```

```
.e-spinner-pane .e-spinner-inner .e-spin-high-contrast .e-path-arc {
stroke: green;
}
`

```

SplitButton

Icons and separator in EJ2 JavaScript Split button control

SplitButton icons

SplitButton can have an icon to provide the visual representation of the action. To place the icon on a SplitButton, set the [iconCss](#) property to `e-icons` with the required icon CSS. By default, the icon is positioned to the left side of the SplitButton. You can customize the icon's position by using the [iconPosition](#) property.

In the following example, the SplitButton with default iconPosition and `iconPosition` as `Top` is showcased:

INDEX.TS

```
import { SplitButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let items: ItemModel[] = [
    {
        text: 'Undo'
    },
    {
        text: 'Redo'
    },
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
//To place the icon in SplitButton.
let splitBtn: SplitButton = new SplitButton({ iconCss: 'e-sb-icons e-paste',
items: items }, '#iconbutton');
let splitBtnObj: SplitButton = new SplitButton({ iconCss: 'e-sb-icons e-
paste', items: items, iconPosition: "Top" }, '#iconpstn');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 SplitButton</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Paste</button>
        <button id="iconpbtn">Paste</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008c8f;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-split-btn-wrapper{
    margin: 20px 20px 5px 5px;
}
@font-face {
    font-family: 'ddb-icons';
    src:

```

```
url(data:application/x-font-ttf; charset=utf-8;base64,AAEAAAAKAIAAwAgTlMvMj0gSRkAAAEoAAAAMVntYXNleDNe+dkAAABlAAAAADxnbHlmh33NQAAAdwAAAJMaGVhZBKOK9sAADQAAAAANmhoZWElHeANwAAAArAAAACRobXR4E6AAAAAAYAAAAUbG9jYQGOAegAAAHQAAAAADG1heHABEWBlAAABCAAAACBuYWllLBM9QAABCgAAAT9cG9zMJntbUAAAZoAAAAUAABAADUV9qAFoEAAAAAAAAADygABAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAAACAACAAAAAAAAAAAAEAFAFkABAAAAAAAGAAAAAoACgAAAP8AAAAAAAAAAAAAQPTAZAABQAAAnoCvAAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWGPsAJYAAAAABAAAAAAAAABAAAAAPoAAA  
D6AAAA+gAAPoAAAAAACAAAAAwAAABQAAWABAAAAFAAEACgAAAAEEAAQAAQAA5wP//wAA5wD//wA  
AAAEABAAAAEAAGADAQAAAAAAI4AwGEAASYAAwAA//oDNQPsaA4AHQBYYAAALHgEOAScmJy4BNz4  
BMzIFFGYHBgcGLgE2NzYzMHYBHgEXDgEHDgIWFfxYXFjY3NjQ3PgE3HgEXFHQXHGE3PgE3PgE  
uAScuAScuAScs+ATc+AQcLASYWAVEffxo6IBkNCQIHcy8bcQG9BwIJDRkgOhoXHwoKGi/+TRIRDyE  
OIxo+ExckFAQMFikwVhcMBWyIFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4caQvEwwsB6gtDTyc  
JCBSsKxYhJ0gWKXIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsakQ4ONznicyYXNBgyNBcmiyc3OA  
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAOqa+kABQANABcAHwAAARUZFSERAYERIzU  
jnSEBIRehesSMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks  
Bdz4//ksB9AF2fHw+Pj8AAAIATAAAAA7cd6QACACQAAAEHEwMOAQcVITUmJyYlND8BIRcWFxYVFAC  
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGgaJxUcLP7PDafNAVL+PHBHCbs  
bBgSUKR8wX3owBg4NFgsQGxsDFxlzAyMAAAACAAAAAAPKA+oAAgATAAABFxEBDgEHhgEXETMRME  
zETM1IQl+zPlabpADA5t0f2F+XP41afbMAZgBJwmYchSBa/48A2r8lgNqfgAAAAASAN4AAQAAAAA  
AAAAABAAAAAQAAAAAAQAJAAEAAQAAAAAAAgAHAaoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA  
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACGsADCAAQAAAAAACWASAGMAAwABBakAAAAACAHU  
AAwABBakAAQASAHCaaWABBakAAgAOAtkaAwABBakAAwAsAJcAAwABBakABAASAKKaAwABBakABQA  
WALSAAwABBakABgASANEAAwABBakACgBYAOMAawABBakAcWAkatSGZGRiLWLjb25zUmVndWxhcmR  
kYilpY29uc2RkYilpY29uc1Zlcncnpb24gMS4wZGRiLWLjb25zRm9udCBnZW5lcmlF0ZWQgdXNpbmc  
gU3luY2Zlc2lvbiBNZXxybyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIAABkAGQAYgAtAGkAYwb  
vAG4AcwBSAGUAzwBlAGwAYQByAGQAZABiAC0AaQBJAG8AbgBzagQAZABiAC0AaQBJAG8AbgBzAFY  
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB  
YAGEadABLAGQAIABIAHMAaQBUAGcAIAbTAHkAbgBjAGYAdQBzagKAbwBuACAATQBlAHQAcbgBvACA  
AUwBOAHUAZABpAG8AdwB3AhcALgBzAHkAbgBjAGYAdQBzagKAbwBuAC4AYwBvAG0AAAAAAgAAAAA  
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbno  
OcGFyYSltYXJrLS0tMDMAAA==) format('trueType');  
font-weight: normal;  
font-style: normal;  
  
.e-sb-icons {  
font-family: 'ddb-icons' !important;  
speak: none;  
font-size: 55px;  
font-style: normal;  
font-weight: normal;  
font-variant: normal;  
text-transform: none;  
line-height: 1;  
-webkit-font-smoothing: antialiased;  
-moz-osx-font-smoothing: grayscale;  
}  
  
.e-paste::before {  
content: '\e701';  
}  
  
.e-cut::before {  
content: '\e700';  
}  
  
.e-copy::before {  
content: '\e70a';  
}
```

The Essential JS 2 provides a set of icons that can be loaded by applying **e-icons** class name to the element.

You can also use third party icons on the splitBtn using the [iconCss](#) property.

Vertical button

Vertical Button in SplitButton can be achieved by adding **e-vertical** class using [cssClass](#) property.

The following example illustrates how to provide vertical support in SplitButton component.

INDEX.TS

```
import { SplitButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let items: ItemModel[] = [
    {
        text: 'Undo'
    },
    {
        text: 'Redo'
    },
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
//To position the icon to the top of the text on a SplitButton.
let splitBtn: SplitButton = new SplitButton({ iconCss: 'e-sb-icons e-paste',
cssClass: 'e-vertical', items: items, iconPosition: 'Top' }, '#iconbutton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Paste</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-split-btn-wrapper{
    margin: 20px 20px 5px 5px;
}
@font-face {
    font-family: 'ddb-icons';
    src:
    url(data:application/x-font-ttf;charset=utf-
    8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDNdE+dkAAABlAAAAADxnbHlmlh3
    3NQAAAdwAAAjMAgVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAAACRobXR4E6AAAAAAAYAAAAA
    UbG9jYQGOAegAAAHQAAAADG1heHABEwBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMJntbU
    AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
    PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAAAAAAAAEAAAAFAFkABAAAAAA
    AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAA
    AAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAAABAAAAAAAAABAAAAAPoAAA
    D6AAAA+gAAAPoAAAAAAACAAAAAwAAABQAawABAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
    AAEEABAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAALHgEOAScmJy4BNz4
    BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
    uAScuAScuASc+ATc+AQcLASYWAVEfFx06IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKgi/+TR1RDyE
    OIxo+ExckFAQMfikwVhCMBwYlFRYkBWCMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
    JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsaKQ40NzcnIyYXNBgYNBcmIyc3OA8
    GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAAQA+KABQANABCAHwAAARUzFSErAYERIZU
    jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
    BdZ4//ksB9AF2fHw+Pj8AAAIAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAC
    GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASStEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBS

```

```

bBgSUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zP1abpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAQAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2l2b25zY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABLAGQAIAB1AHMAaQBvAG4AIAAxAAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnc
OcGFyYS1tYXJrLS0tMDMAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-sb-icons {
  font-family: 'ddb-icons' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
  content: '\e701';
}
.e-cut::before {
  content: '\e700';
}
.e-copy::before {
  content: '\e70a';
}

```

The Essential JS 2 provides a set of icons that can be loaded by applying **e-icons** class name to the element. You can also use third party icons on the SplitButton using the [iconCss](#) property.

Separator

The Separators are the horizontal lines that are used to separate the popup items. You **cannot** select the separators. You can enable separators to group the popup items using the **separator** property.

The following example illustrates how to enable separator support in SplitButton component.

INDEX.TS

```

import { SplitButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let items: ItemModel[] = [
  {
    text: 'Cut',

```

```

        iconCss: 'e-sb-icons e-cut'
    },
    {
        text: 'Copy',
        iconCss: 'e-icons e-copy'
    },
    {
        text: 'Paste',
        iconCss: 'e-sb-icons e-paste'
    },
    {
        separator: true
    },
    {
        text: 'Font',
        iconCss: 'e-sb-icons e-font'
    },
    {
        text: 'Paragraph',
        iconCss: 'e-sb-icons e-paragraph'
    }
    ]];
//To position the separator in SplitButton.
let splitBtn: SplitButton = new SplitButton({ iconCss: 'e-sb-icons e-paste',
items: items }, '#iconbutton');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="iconbutton">Paste</button>
  </div>
<script>
var ele = document.getElementById('container');
```

```
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.e-split-btn-wrapper{
    margin: 20px 20px 5px 5px;
}
@font-face {
font-family: 'ddb-icons';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDNdE+dkAAABlAAAADxnbHlmlh3
3NQAAAdwAAAJMaGVhZBKOK9sAADQAAAAANmhoZWEHeANWAAAArAAAACRobXR4E6AAAAAAAAAYAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEWBlAAABCAAAACBuYWll1lLBM9QAABCgAAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUV9qAFoEAAAAAAAAADygABAAAAAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAAACAACAAAAAAAAAAAAEAFAFkABAAAAAA
AAgAAAAoACgAAP8AAAAAAAAAAAAQPtAZAABQAAAAnoCvAAAAITwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWGPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAPoAAAAAAAACAAAAAWAABQAAWABAAAAFAAEACgAAAAEEAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEEAGADAQAAAAAAI4AwGEAASyAAWAA//oDNQPsAA4AHQBYYAAALHgEOAScmJy4BNz4
BMzIFffGYHBgcGLGeF2nzYZmhYBHGXEDgEHdgEHdgIWFxyXFjY3NjQ3PgE3HgEXFHqXHgE3PgE3PgE
uAScuAScuASc+ATc+AQCLASYWAVEfFxO6IBkNCQIHcy8bCQG9BwIJDRkgOhoxHwoKGi/+TR1RDyE
OIxo+ExckFAQMFikwVhcMBwYlFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKXIaCQknUEILAyCcf2TPi0w2HBUMdg0sOzsakQ4ONzniciYXNBgYNBcmiiyc3OA8
GHRQaoZssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEEEAAAAOqa+kABQANABcaHWAAARUZFSERAYERIzU
jnSEBIRehESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAAIAAAAAA7cd6QACACQAAAEhEwMOAQcVITUmJyYlND8BIRcWFXYVFAC
GKwEVITUmJyYnASMCKP8AguQrOyOBGkIRHREkAsStEGeEDhQxEQGgaJxUCLP7PDafNAVL+PHBHCbs
bBgSUKR8wX3owBg4NFgsQGxsDFxlzAyMAAACAAAAAAPKA+oAagATAAABFXEBdgEHHgEXETMRMXE
zETM1IQL+zPlabpADA5t0f2F+XP41afbMAZgBJwmYCHSBa/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAAQAAAAAAQAUAEEAAQAAAAAAAgAHAaooAAQAAAAAAAwAJABEAAQAAAAABAAJBABoAAQA
AAAAABQALACMAAQAAAAABGAJAC4AAQAAAAAACGAsAdCAAQAAAAAACwASAGMAAwABBakaAAAAACHU
AAwABBakaAQASAhaCaawABBakaAgAOAIkaAwABBakaAwASAJcaAwABBakaABAASAKkaAwABBakaBQA
WALSaAwABBakaBgASANEAAwABBakaCGBYAOMAawABBakaCWakATsgZGRiLWLjb25zUmVndWxhcmR
kyIlpY29uc2RkYilpY29uc1Zlcnpbn24gMS4wZGRiLWLjb25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIAbkAGQAYgAtAgkAYwB
vAG4AcwBSAGUZAzwBlAGWAYQByAGQAZABiAC0AaQBJAG8AbgBzAFY
AZQByAHMAAQBAQ4AIAAAxC4AMABkAGAYgAtAgkAYwBvAG4AcwBGAG8AbgB0ACAAZhwBlAG4AZQB
YAGEADABLAGQYIABLAIHMAAQBAAGCAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBLaHAgBvACA
AUwB0AHUAZABpAG8AdwB3AhcAlGbzAhkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAagAAAAA
```

```

AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnQ
OcGFyYS1tYXJrLS0tMDMAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-sb-icons {
font-family: 'ddb-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-cut::before {
content: '\e700';
}
.e-copy::before {
content: '\e70a';
}
.e-paste::before {
content: '\e701';
}
.e-font::before {
content: '\e702';
}
.e-paragraph::before {
content: '\e703';
}
}

```

See Also

- [SplitButton popup with icons](#)

Popup items in EJ2 JavaScript Split button control

Icons

The Popup action item have an icon or image to provide visual representation of the action. To place the icon on a popup item, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the popup action item.

In the following sample, the icons for Cut, Copy, and Paste menu items are added using the iconCss property.

INDEX.TS

```

import { SplitButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let items: ItemModel[] = [
    {
        text: 'Cut',

```

```

        iconCss: 'e-sb-icons e-cut'
    },
    {
        text: 'Copy',
        iconCss: 'e-icons e-copy'
    },
    {
        text: 'Paste',
        iconCss: 'e-sb-icons e-paste'
    }
  ]];
//To position the icons in SplitButton.
let splitBtn: SplitButton = new SplitButton({ iconCss: 'e-sb-icons e-paste',
items: items }, '#iconbutton');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="iconbutton">Paste</button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}

```

```
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

.e-split-btn-wrapper{
    margin: 20px 20px 5px 5px;
}

@font-face {
font-family: 'ddb-icons';
src:
url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAwAgTlMvMj0gSRkAAAEoAAAAMVntYXDNe+dkAAABlAAAAADxnbHlmh3
3NQAAAdwAAAJMaGVhZBKOK9sAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAAAAAAA
UbG9jYQGOAegAAAAHQAAAADGlheHABEWBlAAABCAAAACBuYWlllLBM9QAABCgAAAI9cG9zdMJnbnU
AAAZoAAAAUAABAAADUV9qAFoEAAAAAAAAADygABAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAAACAACAAAAAAAAAAAAEAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWGPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAWABAAAAFAAEACgAAAAEEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEEAGADAQAAAAAAI4AwGEAAASYAAwAA//oDNQPSAA4AHQBYYAAALHgEOAScmJy4BNz4
BmZIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFHQXHGE3PgE3PgE
uAScuAScuASc+ATc+AQC+LASYWAVEffxo6IBkNCQIHcy8bCQG9BwIJDRkgOhoXHwoKgi/+TR1RDye
OIxo+ExckFAQMFikvVhcMBWyIFRYkBwcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBSsKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsakQ4ONzcniiYXNBgyNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEAAAAAOqa+kABQANABcAHwAAARUZFSERAYERIzU
jnSEBIREhesSMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gf2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAAAA7cd6QACACQAAAEHEwMOAQcVITUmJyY1ND8BIRcWFxYVFAC
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEGeEDhQxEQGgaJxUCLP7PDAFNVL+PHBHCBS
bBgSUKR8wX3owBg4NFgsQGxsDFxlzAyMAAAACAAAAAAPKA+oAAGATAAABFxEBDgEHhgEXETMRMxE
zETM1IQl+zPlabpADA5t0f2F+XP41afbmAZgBJwmYchSBa/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAAQAAAAAAQAJAEEAAQAAAAAAAgAHAaoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABGAJAC4AAQAAAAAACGsAdCAAQAAAAAACWASAGMAAwABBakAAAACAHU
AAwABBakAAQASAHCaaWABBakAAgAOAtkaAwABBakAAwAsAJcAAwABBakABAASAKkaAwABBakABQA
WALSAAwABBakABgASANEAAwABBakACgBYAOMAawABBakAcWakATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1Zlcnpb24gMS4wZWZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc1vbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc1vbi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwBLAGwAYQByAGQAZABiAC0AaQBjAG8AbGBzAGQAZABiAC0AaQBjAG8AbGBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbGB0ACAAZwBLAG4AZQB
yAGEadABLAGQAIABLAHMAaQBUAGCAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBIHQAcgBvACA
AUwBOAHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnoQ
OcGFyYSltYXJrLS0tMDMAAA==) format('truetype');
font-weight: normal;
font-style: normal;
}

.e-sb-icons {
    font-family: 'ddb-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
```

```

text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
  content: '\e701';
}
.e-cut::before {
  content: '\e700';
}
.e-copy::before {
  content: '\e70a';
}

```

Template

Item templating

Popup items can be customized by using the [beforeItemRender](#) event. The item render event triggers while rendering each Popup action item. The event argument will be used to identify the action item and customize it based on the requirement.

In the following example, the icons in each `[li]` items is right aligned by appending span element while `[li]` rendering:

INDEX.TS

```

import { SplitButton, ItemModel, SplitButtonModel, MenuEventArgs } from
 '@syncfusion/ej2-splitbuttons';
import { enableRipple, createElement } from '@syncfusion/ej2-base';
enableRipple(true);
let items: ItemModel[] = [
  {
    text: 'Cut',
  },
  {
    text: 'Copy',
  },
  {
    text: 'Paste',
  }
];
let menuOptions: SplitButtonModel = {
  items: items,
  iconCss: 'e-sb-icons e-paste',
  beforeItemRender: (args: MenuEventArgs) => {
    let shortCutSpan: HTMLElement = createElement('span');
    let text: string = args.item.text;
    args.element.appendChild(shortCutSpan);
    shortCutSpan.setAttribute('class', 'shortcut');
    let clsName: string = (text === 'Copy') ? 'e-icons' : 'e-sb-
icons';
    shortCutSpan.classList.add(clsName);
    (text === 'Cut') ? shortCutSpan.classList.add('e-cut') : (text
=== 'Paste') ? shortCutSpan.classList.add('e-paste') :
shortCutSpan.classList.add('e-copy');

```



```

    }
    };
    let splitBtn: SplitButton = new SplitButton(menuOptions, '#action');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="action"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
.shortcut {
  float: right;
  margin-top: 9px;
  padding-left: 30px;
}
#loader {
  color: #008c8f;
  height: 40px;
  left: 45%;
}

```

Copyright © 2001 -2024 Syncfusion Inc. 1318

```
.e-cut::before {
  content: '\e700';
}
.e-copy::before {
  content: '\e70a';
}
.e-split-btn-wrapper{
  margin: 20px 20px 5px 5px;
}
```

Popup templating

The whole popup can be customized as per the requirement and it can be customized by handling it in [target](#) property.

In the following sample, the whole popup item is customized as color palette by giving `div` as target and it can be achieved using `target` property.

INDEX.TS

```
import { SplitButton, SplitButtonModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let menuOptions: SplitButtonModel = {
  target: '#dropdowntarget',
  iconCss: 'e-sb-icons e-color',
};
let splitBtn: SplitButton = new SplitButton(menuOptions, '#action');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
  <button id="action"></button>
  <div id="dropdowntarget">
    <div id="first">
      <div id="black"></div>
      <div id="red"></div>
      <div id="green"></div>
      <div id="gray"></div>
      <div id="blue"></div>
      <div id="violet"></div>
      <div id="brown"></div>
      <div id="darkgoldenrod"></div>
      <div id="aquamarine"></div>
    </div>
  </div>
</div><script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
.shortcut {
  float: right;
  margin-top: 9px;
  padding-left: 30px;
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
@font-face {
  font-family: 'paint';
  src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMj0gSRIAAAEoAAAAVmNtYXNlEODVAAABiAAAADZnbHlmIZD
+uwAAAcgAAADMaGVhZBKhHQAADQAAAAANmhoZWEHjANrAAAArAAAACRobXR4B+j/8wAAAYAAAAA
IbG9jYQBMAAAAAHAAAAABmlheHABDgBKAAABCAAAACBuYW1ln6hzswAAApQAAAINcG9zdEkLMmU
AAASkAAAAANgABAAADUv9qAFoEAP/z//4D6gABAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAAZfc6F8
PPPUACwPoAAAAANfSn9kAAAAA19Kf2f/z//wD6gPhAAAAACAACAAAAAAAAAAAAEAAAAACAD4AAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPhAJYAAAABAAAAAAAAABAAAAAPo//M
AAAACAAAAAwAAABQAawABAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAZgAAAL/8//8A+oD4QAKAD0AAAEWBgceATc1JiQHJTMmNjceARcVJx4BBx4BFQ4BIiYnNDY
3PgEvAS4BIw4BBwEGHgI3AT4BLwE1LgEnDgEDEiRlCgulCxP+8RT+GyYDQFxoZQwTBQEDDxEbJzo

```

```

nAREOCQkPJQ4cDBcdAf6oG1a3nx8BWQ4RHKAdeG1oWwHTLHVwYVmL6Kx1BHEqfwYFqWUHEx4tDAo
cEx0nJx0RHgoVUDQpDgsBFAH+px2guFUaAVkNoiCgCXnhCAWOAAAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAAEABQABAAEAAAAAAAAIABwAGAAEAAAAAAAAAMABQANAAEAAAAAAAAQABQASAAEAAAAAAAAU
ACwAXAAEAAAAAAAAAYABQAiAAEAAAAAAAAoALAAEAAAAAAAAAsAEgBTAAMAAQQJAAAAAgB1AAMAAQQ
JAAEACgBnAAMAAQQJAAIADgBxAAMAAQQJAAMACgB/AAMAAQQJAAQACgCJAAMAAQQJAAUAFgCTAAM
AAQQJAAYACgCpAAMAAQQJAAoAWACzAAMAAQQJAAsAJAELIHBhaW50UmVndWxhcncBhaW50cGFpbmR
WZXJzaW9uIDEuMHBhaW50Rm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIAAwAGEAaQBwAHQAUGBlAGcAdQBsAGEAcgBwAGEAaQBwAHQ
AcABhAGkAbgB0AFYAZQByAHMAaQBvAG4AIAAxAC4AMABwAGEAaQBwAHQAQgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdAB1AGQAAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMADHBhaW50LWJ1Y2tldAAAAA=)
format('truetype');
font-weight: normal;
font-style: normal;
}
.e-sb-icons {
  font-family: 'paint' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-color::before {
  content: '\e700';
  color: black;
}

.e-split-btn-wrapper{
  margin: 20px 20px 5px 5px;
}
#dropdowntarget {
  border: 0.5px solid grey;
  height: 110px;
  width: 110px;
}

}
#black {
  width: 30px;
  height: 30px;
  background-color: black;
  margin: 5px 5px;
  float: left;
}
#red{
  width: 30px;
  height: 30px;
  background-color: red;
  margin: 5px 0px;
  float: left;
}
}

```

```
#green{
    width: 30px;
    height: 30px;
    background-color: green;
    margin: 5px 5px;
    float:left;
}
#gray{
    width: 30px;
    height: 30px;
    background-color: gray;
    margin: 0px 5px;
    float:left;
}
#blue{
    width: 30px;
    height: 30px;
    background-color: blue;
    float:left;
}
#violet{
    width: 30px;
    height: 30px;
    background-color: violet;
    margin: 0px 5px;
    float:left;
}
#brown{
    width: 30px;
    height: 30px;
    background-color: brown;
    margin: 5px 5px;
    float:left;
}
#darkgoldenrod{
    width: 30px;
    height: 30px;
    background-color: darkgoldenrod;
    margin: 5px 0px;
    float:left;
}
#aquamarine{
    width: 30px;
    height: 30px;
    background-color: aquamarine;
    margin: 5px 5px;
    float:left;
}
#icon{
    width: 10px;
    height: 10px;
    background-color: aquamarine;
    position: absolute;
}
```

See Also

- [Popup items grouping](#)
- [SplitButton popup with separator](#)

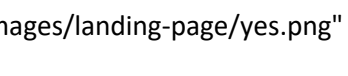
Accessibility in EJ2 JavaScript Split button control

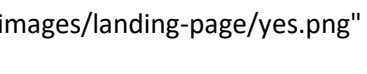
The Split button component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

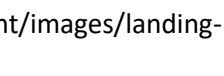
The accessibility compliance for the Split button component is outlined below.

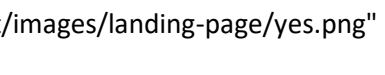
| Accessibility Criteria | Compatibility |

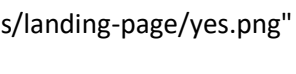
| -- | -- |

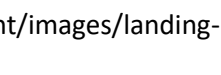
| [WCAG 2.2](#) Support |  |

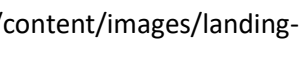
| [Section 508](#) Support |  |

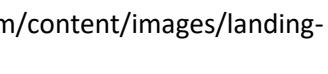
| Screen Reader Support |  |


| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Split button component followed the [WAI-ARIA] patterns to meet the accessibility. The following ARIA attributes are used in the Split button component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Split button component as **button**, Split button popup as **menu**, and the Split button popup action items as **menuitem**. |

| **aria-haspopup** | Indicates the availability and type of interactive Split button popup element. |

| **aria-expanded** | Indicates whether the Split button popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Split button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Split button component.

| Press | To do this |

| --- | --- |

| **Esc** | Closes the opened popup. |

| **Enter** | Opens the popup, or activates the highlighted item and closes the popup. |

| **Spacer** | Opens the popup. |

| **Up** | Navigates up or to the previous action item. |

| **Down** | Navigates down or to the next action item. |

| **Alt + Up Arrow** | Closes the popup. |

| **Alt + Down Arrow** | Opens the popup. |

Ensuring accessibility

The Split button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Split button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Split button component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Create right to left splitbutton in EJ2 JavaScript Split button control

SplitButton component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in SplitButton component.

INDEX.TS

```
import { SplitButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let items: ItemModel[] = [
    {
        text: 'Autosum'
    },
    {
        text: 'Average'
    },
    {
        text: 'Count numbers',
    },
    {
        text: 'Min'
    },
    {
        text: 'Max'
    }
];
//To enable RTL in SplitButton.
let splitBtn: SplitButton = new SplitButton({ iconCss: 'e-sb e-sigma',
items: items, enableRtl: true }, '#iconbutton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="styles.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Autosum</button>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
    font-family: 'sb-icon';
    src:
    url(data:application/x-font-ttf;charset=utf-
    8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXDNdEOdVAAABiAAAADZnbHlmFjF
    e0gAAAcgAAABIAgVhZBIxlUcAAADQAAAAANmhoZWEHhANTAAAArAAAACRobXR4B+gAAAAAYAAAAA
    IbG9jYQAAkAAAAAHAAAAABmlheHABDQAAeAAABCAAAACBuYW11akQFAwAAAhAAAAIILcG9zdEP61+c
    AAAQ4AAAAAMwABAAADUv9qAFoEAAAAAADbgABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAAj4iO918
    PPPUACwPoAAAAANfSgDwAAAAA19KoPAAAAAADbgPqAAAAACAACAAAAAAAAAAAAEAAAACABIAAQAAAAA
    AAgAAAAoACgAAAP8AAAAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
    AAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
    AAAACAAAAAwAAABQAAwABAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAAA
    AAAAAJAAAAAEAAAAAA24D6gARAAATHQETARUhEyMHlQEDIRUzESG65f7gAsolLBH9ywES9gIFLfl
    wA5xDQv6u/pBSAQNyAVwBaXIBBAAAAAAAAAEgDeAAEAAAAAAAAAAAAQAAAAEAAAAAAAAEABwABAAEAAAA
    AAAIABwAIAAEAAAAAAAAAMABwAPAAEAAAAAAAAAQABwAWAAEAAAAAAAAAUACwAdAAEAAAAAAAAAYABwAoAAE
    AAAAAAaALAAvAAEAAAAAAAAAsAEgBbAAMAAQQAIAAAAgBtAAMAAQQAIAAEADgBvAAMAAQQAIAAIADgB
    9AAMAAQQAIAAMADgCLAAMAAQQAIAAQADgCZAAMAAQQAIAAUAFgCnAAMAAQQAIAAYADgC9AAMAAQQAIAAo
    AWADLAAMAAQQAIAAsAJAEjIHNiLW1jb25SZWd1bGFYc2ItaWNvbnNiLW1jb25WZXJzaW9uIDEuMHN
    iLW1jb25Gb250IGd1bmVYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
    jZnVzaW9uLmNvbQAgAHMAYgAtAGkAYwBvAG4AUgBlAGcAdQBsAGEAcgBzAGIALQBpAGMABwBuAHM

```

```

AYgAtAGkAYwBvAG4AVgBlAHIAcwBpAG8AbgAgADEALgAwAHMAYgAtAGkAYwBvAG4ARgBvAG4AdAA
gAGcAZQBUAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0
AZQB0AHIAbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMabwB
tAAAAAIAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMACXN1bW1hdG1vbGAAAA=
=) format('truetype');
font-weight: normal;
font-style: normal;
}
.e-sb {
  font-family: 'sb-icon' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-sigma::before {
  content: '\e700';
}
.e-split-btn-wrapper{
  margin: 20px 20px 5px 5px;
}

```

Group items in popup in EJ2 JavaScript Split button control

Items in popup can be grouped in SplitButton by templating entire popup with ListView. To achieve grouping in ListView, check [ListView Grouping](#) documentation. To template ListView in popup, create ListView with id `listview` and provide it as `target` for SplitButton.

The following example illustrates how to group items in popup using ListView component.

INDEX.TS

```

import { SplitButton } from '@syncfusion/ej2-splitbuttons';
import { ListView } from '@syncfusion/ej2-lists';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let listItems: { [key: string]: Object }[] = [
  {
    'text': 'Cut',
    'category': 'Basic'
  },
  {
    'text': 'Copy',
    'category': 'Basic'
  },
  {
    'text': 'Paste',
    'category': 'Basic'
  },
  {
    'text': 'Paste as Formula',
    'category': 'Advanced'
  }
]

```

```

    },
    {
        'text': 'Paste as Hyperlink',
        'category': 'Advanced'
    },
];
let listView: ListView = new ListView({
    dataSource: listItems,
    sortOrder: 'Descending',
    fields: { groupBy: 'category' }
}, '#listview');
let splitBtn: SplitButton = new SplitButton({ content: 'ClipBoard', target:
listView.element }, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SplitButton</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="element"></button>
        <div id="listview"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Open a dialog on popup item click in EJ2 JavaScript Split button control

This section explains about how to open a dialog on SplitButton popup item click. This can be achieved by handling dialog open in [select](#) event of the SplitButton.

In the following example, Dialog will open while selecting **Update...** item.

INDEX.TS

```
import { SplitButton, ItemModel, MenuEventArgs } from '@syncfusion/ej2-splitbuttons';
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let dialogObj: Dialog = new Dialog({
    width: '250px',
    header: 'Software Update',
    content: 'Are you sure want to update?',
    target: document.getElementById('container'),
    visible: false,
    buttons: [
        {
            click: () => {
                dialogObj.hide();
            },
            buttonModel: { content: 'OK', isPrimary: true }
        },
        {
            click: () =>{
                dialogObj.hide();
            },
            buttonModel: { content: 'Cancel', isPrimary: true }
        }
    ],
});
dialogObj.appendTo('#dialog');
let items: ItemModel[] = [
    {
        text: 'Help'
    },
    {
        text: 'About'
    }
];
```

```

    },
    {
        text: 'Update...'
    }
];
let splitBtn: SplitButton = new SplitButton(
{
    content: 'Help',
    items: items,
    select: (args: MenuEventArgs) => {
        if (args.item.text === 'Update...') {
            dialogObj.show();
        }
    }
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="element"></button>
    <div id="dialog"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
    visibility: hidden;
}
html,
body,
#container {
    height: 100%;
    overflow: hidden;
    width: 100%;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Set the disabled state in EJ2 JavaScript Split button control

SplitButton component can be enabled or disabled using [disabled](#) property. To disable SplitButton component, set the disabled property as `true`.

The following example illustrates how to set the disable state in SplitButton component.

INDEX.TS

```
import { SplitButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let items: ItemModel[] = [
    {
        text: 'Autosum'
    },
    {
        text: 'Average'
    },
    {
        text: 'Count numbers',
    },
    {
        text: 'Min'
    },
    {
        text: 'Max'
    }
];
//To disable the SplitButton.
let splitBtn: SplitButton = new SplitButton({ iconCss: 'e-sb e-sigma',
items: items, disabled: true }, '#iconbutton');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```
<title>EJ2 SplitButton</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="styles.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <button id="iconbutton">Autosum</button>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008c9f;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
font-family: 'sb-icon';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAVMnTYXDnEOdVAAABiAAAADZnbHlMfJF
e0gAAACgAAABIAgVhZBIxlUcAAADQAAAAANmhoZWEHHANtAAAArAAAACRobXR4B+gAAAAAYAAAAA
IbG9jYQAKAAAAAAHAAAAABm1heHABDQAAeAAABCAAAACBuYw1lakQFAwAAAhAAAAIlcG9zdEP61+c
AAQ4AAAAAMwABAAADUv9qAFoEAAAAAADbqABAAAAAAAAAAAAAAAAAAAAqABAAAAAQAAj4iO918
```



```

PPPUACwPoAAAAANfSqDwAAAAA19KoPAAAAAADbgPqAAAACAACAAAAAAAAAAAAEAAAACABIAAQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
AAAACAAAAAwAAABQAAwABAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAJAAAAEAAAAAA24D6gARAAATHQETARUHEyMHIQEDIRUzESG65f7gAsolLBH9ywES9gIFLfl
wA5xDQv6u/pBSAQNyAVwBaXIBBAAAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEABwABAAEAAAA
AAAIABwAIAAEAAAAAAAAAMABwAPAAEAAAAAAAAAQABwAWAAEAAAAAAAAAUACwAdAAEAAAAAAAAAYABwAoAAE
AAAAAAoALAAvAAEAAAAAAAAsAEgBbAAMAAQQJAAAAAgBtAAMAAQQJAAEADgBvAAMAAQQJAAIADgB
9AAMAAQQJAAAMADgCLAAMAAQQJAAQADgCZAAMAAQQJAAUAFgCnAAMAAQQJAAAYADgC9AAMAAQQJAAo
AWADLAAMAAQQJAAAsAJAEjIHNiLWlj25SZWd1bGFyc2ItaWNvbnNiLWlj25WZXJzaW9uIDEuMHN
iLWlj25Gb250IGdlbmVYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAHMAYgAtAGkAYwBvAG4AUgBlAGcAdQBsaGEAcgBzAGIALQBpAGMabwBuAHM
AYgAtAGkAYwBvAG4AVgBlAHIAcWBPAG8AbgAgADEALgAwAHMAYgAtAGkAYwBvAG4ARgBvAG4AdAA
gAGcAZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0
AZQB0AHIAbwAgAFMAdABlAGQAAQBVaHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0
tAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMACXNlbWlj25WZXJzaW9u
=) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-sb {
  font-family: 'sb-icon' !important;
  speak: none;
  font-size: 55px;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
.e-sigma::before {
  content: '\e700';
}
.e-split-btn-wrapper{
  margin: 20px 20px 5px 5px;
}

```

Underline a character in a text in EJ2 JavaScript Split button control

Underline a particular character in a text can be handled in [beforeItemRender](#) event by adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

In the following example, **C** is underlined in the text **Copy**.

INDEX.TS

```

import { SplitButton, ItemModel, MenuEventArgs } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Initialize action items.
let items: ItemModel[] = [
  {
    text: 'Cut'
  },

```

```

    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
  ];
let splitBtn: SplitButton = new SplitButton({
  content: 'Paste',
  items: items,
  beforeItemRender: (args: MenuEventArgs) => {
    if (args.item.text === 'Copy') {
      args.element.innerHTML = '<u>C</u>opy';
    }
  }
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <button id="element"></button>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Ej1 api migration in EJ2 JavaScript Split button control

This article describes the API migration process of SplitButton component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Specifies the text of the SplitButton | **Property:** *text*

 \$("#splitbutton").ejSplitButton({
target: "#target",
 text: "Split Button"
}); | **Property:** *content*

 var splitButton = new ej.splitbuttons.SplitButton({
 content: "Split Button"
});
splitButton.appendTo("#splitbutton"); |

| Popup content | **Property:** *target*

 \$("#splitbutton").ejSplitButton({
 target: "#target",
 text: "Split Button"
}); | **Property:** *target*

 var splitButton = new ej.splitbuttons.SplitButton({
target: "#target",
 content: "Split Button"
});
splitButton.appendTo("#splitbutton"); |

| Popup items | Not applicable | **Property:** *items*

 var splitButton = new ej.splitbuttons.SplitButton({
 items: items,
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");
 var items = [
 {
 text: 'Cut'
 },
 {
 text: 'Copy'
 },
 {
 text: 'Paste'
 }
]; |

| Arrow position | **Property:** *arrowPosition*

 \$("#splitbutton").ejSplitButton({
arrowPosition: ej.ArrowPosition.Left,
 target: "#target",
 text: "Split Button"
}); | Not applicable |

| Button mode | **Property:** *buttonMode*

 \$("#splitbutton").ejSplitButton({
buttonMode: ej.ButtonMode.Dropdown,
 target: "#target",
 text: "Split Button"
}); | Not applicable |

| Content type | **Property:** *contentType*

 \$("#splitbutton").ejSplitButton({
contentType: ej.ContentType.TextOnly,
text:"login",
 target: "#target"
}); | Not applicable |

| Adding custom css class | **Property:** *cssClass*

 \$("#splitbutton").ejSplitButton({
cssClass: "e-custom-class",
 target: "#target",
 text: "Split Button"
}); | **Property:** *cssClass*

 var splitButton = new ej.splitbuttons.SplitButton({
cssClass: "e-custom-class",
 items: items,
 content: "Split Button"
});
splitButton.appendTo("#splitbutton"); |

| Disabled state | **Property:** *enabled*

 `$("#splitbutton").ejSplitButton({
enabled: false,
 target: "#target",
 text: "Split Button"
});` | **Property:** *disabled*

 `var splitButton = new ej.splitbuttons.SplitButton({
disabled: true,
 items: items,
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");` |

| RTL | **Property:** *enableRTL*

 `$("#splitbutton").ejSplitButton({
enableRTL: true,
 target: "#target",
 text: "Split Button"
});` | **Property:** *enableRtl*

 `var splitButton = new ej.splitbuttons.SplitButton({
enableRtl: true,
 items: items,
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");` |

| Height | **Property:** *height*

 `$("#splitbutton").ejSplitButton({
height: 30,
 target: "#target",
 text: "Split Button"
});` | Not applicable |

| Width | **Property:** *width*

 `$("#splitbutton").ejSplitButton({
width: 100,
 target: "#target",
 text: "Split Button"
});` | Not applicable |

| HTML attributes | **Property:** *htmlAttributes*

 `$("#splitbutton").ejSplitButton({
htmlAttributes : { disabled:"disabled" },
 target: "#target",
 text: "Split Button"
});` | Not applicable |

| Icons | **Property:** *prefixIcon*

 `$("#splitbutton").ejSplitButton({
prefixIcon:"e-icon e-handup",
contentType: ej.ContentType.TextAndImage,
target: "#target",
 text: "Split Button"
});` | **Property:** *iconCss*

 `var splitButton = new ej.splitbuttons.SplitButton({
iconCss: 'e-icons e-handup',
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");` |

| Icon position | **Property:** *imagePosition*

 `$("#splitbutton").ejSplitButton({
imagePosition: ej.ImagePosition.ImageTop,
 prefixIcon:"e-icon e-handup",
contentType: ej.ContentType.TextAndImage,
target: "#target",
 text: "Split Button"
});` | **Property:** *iconPosition*

 `var splitButton = new ej.splitbuttons.SplitButton({
iconPosition: "Top",
 iconCss: 'e-icons e-handup',
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");` |

| Secondary icon | **Property:** *suffixIcon*

 `$("#splitbutton").ejSplitButton({
suffixIcon:"e-icon e-icon-closed",
prefixIcon:"e-icon e-handup",
contentType: ej.ContentType.ImageTextImage,
target: "#target",
 text: "Split Button"
});` | Not applicable |

| Show rounded corner | **Property:** *showRoundedCorner*

 `$("#splitbutton").ejSplitButton({
showRoundedCorner: true,
 target: "#target",
 text: "Split Button"
});` | Not applicable |

| Size | **Property:** *size*

 `$("#splitbutton").ejSplitButton({
size: ej.ButtonSize.Small,
 target: "#target",
 text: "Split Button"
});` | **Property:** *cssClass*

 `var splitButton = new ej.splitbuttons.SplitButton({
cssClass: "e-small",
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");` |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Destroy method | **Method:** *destroy*

 `$("#splitbutton").ejSplitButton({
 target: "#target",
 text: "Split Button"
});
var splitButton =`

`$("#splitbutton").data("ejSplitButton");` `
` `splitButton.destroy();` | **Method:** *destroy* `

` `var splitButton = new ej.splitbuttons.SplitButton({` `
` `content: "Split Button"
});` `
` `splitButton.appendTo("#splitbutton");` `
` `splitButton.destroy();` |

| Disable method | **Method:** *disable* `

` `$("#splitbutton").ejSplitButton({` `
` `target: "#target",` `
` `text: "Split Button"
});` `
` `var splitButton =` `$("#splitbutton").data("ejSplitButton");` `
` `splitButton.disable();` | Not applicable |

| Enable method | **Method:** *enable* `

` `$("#splitbutton").ejSplitButton({` `
` `target: "#target",` `
` `text: "Split Button"
});` `
` `var splitButton = $("#splitbutton").data("ejSplitButton");` `
` `splitButton.enable();` | Not applicable |

| Hide popup | **Method:** *hide* `

` `$("#splitbutton").ejSplitButton({` `
` `target: "#target",` `
` `text: "Split Button"
});` `
` `var splitButton = $("#splitbutton").data("ejSplitButton");` `
` `splitButton.hide();` | **Method:** *toggle* `

` `var splitButton = new ej.splitbuttons.SplitButton({` `
` `items: items,` `
` `content: "Split Button"
});` `
` `splitButton.appendTo("#splitbutton");` `
` `splitButton.toggle();` |

| Show popup | **Method:** *show* `

` `$("#splitbutton").ejSplitButton({` `
` `target: "#target",` `
` `text: "Split Button"
});` `
` `var splitButton = $("#splitbutton").data("ejSplitButton");` `
` `splitButton.show();` | **Method:** *toggle* `

` `var splitButton = new ej.splitbuttons.SplitButton({` `
` `items: items,` `
` `content: "Split Button"
});` `
` `splitButton.appendTo("#splitbutton");` `
` `splitButton.toggle();` |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| BeforeOpen event | **Event:** *beforeOpen* `

` `$("#splitbutton").ejSplitButton({` `
` `beforeOpen: function(args) { / code block / },` `
` `target: "#target",` `
` `text: "Split Button"
});` | **Event:** *beforeOpen* `

` `var splitButton = new ej.splitbuttons.SplitButton({` `
` `beforeOpen: function(args) { / code block / },` `
` `items: items,` `
` `content: "Split Button"
});` `
` `splitButton.appendTo("#splitbutton");` |

| Click event | **Event:** *click* `

` `$("#splitbutton").ejSplitButton({` `
` `click: function(args) { / code block / },` `
` `target: "#target",` `
` `text: "Split Button"
});` | **Event:** *click* `

` `var splitButton = new ej.splitbuttons.SplitButton({` `
` `click: function(args) { / code block / },` `
` `items: items,` `
` `content: "Split Button"
});` `
` `splitButton.appendTo("#splitbutton");` |

| Close event | **Event:** *close* `

` `$("#splitbutton").ejSplitButton({` `
` `close: function(args) { / code block / },` `
` `target: "#target",` `
` `text: "Split Button"
});` | **Event:** *close* `

` `var splitButton = new ej.splitbuttons.SplitButton({` `
` `close: function(args) { / code block / },` `
` `items: items,` `
` `content: "Split Button"
});` `
` `splitButton.appendTo("#splitbutton");` |

| Create event | **Event:** *create* `

` `$("#splitbutton").ejSplitButton({` `
` `create: function(args) { / code block / },` `
` `target: "#target",` `
` `text: "Split Button"
});` | **Event:** *created* `

` `var splitButton = new ej.splitbuttons.SplitButton({` `
` `created: function() { / code block / },` `
` `items: items,` `
` `content: "Split Button"
});` `
` `splitButton.appendTo("#splitbutton");` |

| Destroy event | **Event:** *destroy* `

` `$("#splitbutton").ejSplitButton({` `
` `destroy: function(args) { / code block */ },` `
` `target: "#target",` `
` `text: "Split Button"
});` | Not applicable |

| ItemMouseOut event | **Event:** *itemMouseOut*

 \$("#splitbutton").ejSplitButton({

itemMouseOut: function(args) { / code block */ },
 target: "#target",
 text: "Split Button"

}); | Not applicable |

| ItemMouseOver event | **Event:** *itemMouseOver*

 \$("#splitbutton").ejSplitButton({

itemMouseOver: function(args) { / code block */ },
 target: "#target",
 text: "Split
Button"
}); | Not applicable |

| Item select event | **Event:** *itemSelected*

 \$("#splitbutton").ejSplitButton({

itemSelected: function(args) { / code block / },
 target: "#target",
 text: "Split Button"

}); | **Event:** *select*

 var splitButton = new ej.splitbuttons.SplitButton({
select:
function(args) { / code block / },
target: "#target",
 content: "Split Button"
});

splitButton.appendTo("#splitbutton");|

| Open event | **Event:** *open*

 \$("#splitbutton").ejSplitButton({
open: function(args) { /
code block / },
 target: "#target",
 text: "Split Button"
}); | **Event:** *open*

 var
splitButton = new ej.splitbuttons.SplitButton({
open: function(args) { / code block / },
 items:
items,
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");|

| BeforeClose event | Not applicable | **Event:** *beforeClose*

 var splitButton = new
ej.splitbuttons.SplitButton({
beforeClose: function(args) { / code block */ },
 items: items,

 content: "Split Button"
});
splitButton.appendTo("#splitbutton");|

| BeforeItemRender event | Not applicable | **Event:** *beforeItemRender*

 var splitButton = new
ej.splitbuttons.SplitButton({
beforeItemRender: function(args) { / code block */ },
 items:
items,
 content: "Split Button"
});
splitButton.appendTo("#splitbutton");|

Splitter

Pane sizing in EJ2 JavaScript Splitter control

Splitter allows you to provide pane sizes either in pixel or percentage formats.

Auto size panes

The splitter's panes are adjusted automatically during resizing if the size is not specified externally to panes, because the panes are designed based on flex layout by default. When add/remove or show/hide the panes, the panes are auto aligned within its container.

INDEX.TS

```
import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
  height: '200px',
  width: '600px'
});
splitObj.appendTo('#splitter');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the splitter-->
        <div id="splitter">
            <div>
                <div class="content">
                    <h3 class="h3">Grid </h3>
                    The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">Schedule </h3>
                    The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">Chart </h3>
                    ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
                </div>
            </div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Fixed pane

You can render the split panes with fixed size in both **horizontal** and **vertical** mode. Even if you provide fixed sizes to all the panes, the last pane is considered as a flexible pane.

INDEX.TS

```
import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
    height: '200px',
    paneSettings: [
        { size: '200px' },
        { size: '200px' },
        { size: '200px' }
    ],
    width: '600px'
});
splitObj.appendTo('#splitter');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="splitter">
      <div>Left pane</div>
      <div>Middle pane</div>
      <div>Right pane</div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

INDEX.TS

```
import { Splitter } from '@syncfusion/ej2-layouts';
```



```
let splitObj: Splitter = new Splitter({
  height: '200px',
  paneSettings: [
    { size: '30%' },
    { size: '40%' },
    { size: '30%' }
  ],
  width: '600px'
});
splitObj.appendTo('#splitter');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="splitter">
      <div>Left pane</div>
      <div>Middle pane</div>
      <div>Right pane</div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Pane content in EJ2 JavaScript Splitter control

This section explains how to provide plain text content or HTML markup or integrate other JavaScript UI controls as content of splitter.

HTML Markup

Splitter is a layout based container control. You can render the pane contents from existing HTML markups. Converting HTML markup as splitter pane is easy way to add the panes content dynamically.

INDEX.TS

```
import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
    height: '200px',
    paneSettings: [
        { size: '200px',
          content: '<div class="content"><h3 class="h3">PARIS </h3>Paris, the
city of lights and love - this short guide is full of ideas for how to make
the most of the romanticism...</div>'
        },
        { size: '200px',
          content: '<div class="content"><h3 class="h3">CAMEMBERT </h3>The
village in the Orne département of Normandy where the famous French cheese
is originated from.</div>'
        },
        { size: '200px',
          content: '<div class="content"><h3 class="h3">GRENOBLE </h3>The
capital city of the French Alps and a major scientific center surrounded by
many ski resorts, host of the Winter Olympics in 1968.</div>'
        }
    ],
    width: '600px'
});
splitObj.appendTo('#splitter');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
```

```

        <div id="splitter">
            <div></div>
            <div></div>
            <div></div>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Also, you can provide the pane content inner HTML.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
    height: '200px',
    paneSettings: [
        { size: '200px' },
        { size: '200px' },
        { size: '200px' }
    ],
    width: '600px'
});
splitObj.appendTo('#splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Splitter </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Uploader Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">

```

```

<!--element which is going to render the splitter-->
<div id="splitter">
  <div>
    <div class="content">
      <h3 class="h3">PARIS </h3>
      Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
    </div>
  </div>
  <div>
    <div class="content">
      <h3 class="h3">CAMEMBERT </h3>
      The village in the Orne département of Normandy where
the famous French cheese is originated from.
    </div>
  </div>
  <div>
    <div class="content">
      <h3 class="h3">GRENOBLE </h3>
      The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
    </div>
  </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

JavaScript UI controls

You can render any JavaScript UI controls along with their native and control events within splitter as pane content.

You can refer [Accordion within splitter](#) and [Listview within splitter](#) examples.

Plain content

You can add the plain text as a pane contents using either inner HTML or [content](#) API

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
  height: '200px',
  paneSettings: [
    { size: '200px', content: 'Left pane' },
    { size: '200px', content: 'Middle pane' },
    { size: '200px', content: 'Right pane' }
  ]
});

```

```

    ],
    width: '600px'
  });
  splitObj.appendTo('#splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="splitter">
      <div></div>
      <div></div>
      <div></div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Pane content using selector

You can set HTML element as pane [content](#) using the query selectors such as ID or CSS class selectors.

The following code demonstrates how to fetch an element from the document and load it to pane using its ID.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
/**

```

```

* Splitter pane content as html element using ID selector
*/
let splitObj1: Splitter = new Splitter({
    height: '200px',
    paneSettings: [
        { size: '25%', min: '60px', content: '#left-pane-content' },
        { size: '50%', min: '60px', content: '#middle-pane-content' },
        { size: '25%', min: '60px', content: '#last-pane-content' }
    ],
    width: '100%',
    separatorSize: 4
});
splitObj1.appendTo('#horizontal');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Splitter </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Uploader Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the splitter-->
        <div id="target" class="control-section">
            <!-- Splitter control target -->
            <div id="horizontal"></div>
        </div>

        <!-- pane contents -->
        <div id="left-pane-content" style="display: none;">
            <div>Left pane<div id="panetext">size: 25%</div>
                <div id="panetext">min: 60px</div>
            </div>
        </div>

        <div id="middle-pane-content" style="display: none;">
            <span>Middle pane<div id="panetext">size: 50%</div>
                <div id="panetext">min: 60px</div>
            </span>
        </div>
    </div>

```

```

        </div>

        <div id="last-pane-content" style="display: none;">
            <span>Right pane<div id="panetext">size: 25%</div>
            <div id="panetext">min: 60px</div>
        </span>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Split panes in EJ2 JavaScript Splitter control

This section explain about split panes behaviours.

Horizontal layout

By default, splitter will render in horizontal orientation. Splitter container will be splitted as panes in horizontal flow direction with vertical seperator.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
// Initialize Splitter control
let splitObject: Splitter = new Splitter({
    height: '250px',
    paneSettings: [
        { size: '200px' },
        { size: '200px' },
        { size: '200px' }
    ],
    width: '600px'
});
// Render initialized Splitter
splitObject.appendTo('#splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Splitter </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Uploader Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the splitter-->
        <div id="splitter">
            <div>
                <div class="content">
                    <h3 class="h3">Grid </h3>
                    The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">Schedule </h3>
                    The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">Chart </h3>
                    ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Vertical layout

By setting [orientation](#) API as **Vertical**, splitter will render in vertical orientation. Splitter container will be splitted as panes in vertical flow direction with horizontal separator.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
// Initialize Splitter control
let splitObject: Splitter = new Splitter({
    height: '305px',
    paneSettings: [
        { size: '100px' },
        { size: '100px' },
    ]
});

```



```

        { size: '100px' }
    ],
    width: '600px',
    orientation: 'Vertical'
});
// Render initialized Splitter
splitObject.appendTo('#splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Splitter </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Uploader Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the splitter-->
        <div id="splitter">
            <div>
                <div class="content">
                    <h3 class="h3">Grid</h3>
                    The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">Schedule</h3>
                    The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">Chart</h3>
                    ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
                </div>
            </div>
        </div>
    </div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can also render multiple panes in splitter with both **Horizontal/Vertical** orientations.

Separator

By default, pane separator will be render with **1px** width/height. You can customize the separator size by using [separatorSize](#) API.

For horizontal orientation, it will be considered as separator width.

For vertical orientation, it will be considered as separator height.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
// Initialize Splitter control
let splitObject: Splitter = new Splitter({
    height: '250px',
    paneSettings: [
        { size: '200px' },
        { size: '200px' },
        { size: '200px' }
    ],
    width: '600px',
    separatorSize: '5'
});
// Render initialized Splitter
splitObject.appendTo('#splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Splitter </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Uploader Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="splitter">
      <div>
        <div class="content">
          <h3 class="h3">Grid </h3>
          The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">Schedule </h3>
          The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">Chart </h3>
          ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
        </div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Nested Splitter

Splitter provides support to render the nested pane to achieve the complex layouts. You can use the same `<div>` element for splitter pane and nested splitter.

Also you can render the nested splitter using direct child of the splitter pane. For this, nested splitter should have 100% width and height to match with the parent pane dimensions.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
let horizontalObj: Splitter = new Splitter({
  height: '316px',
  paneSettings: [
    { size: '200px' },
    { size: '200px' },
    { size: '200px' }
  ],
},

```

```

        width: '600px'
    });
    horizontalObj.appendTo('#Horizontal_splitter');
    let verticalObj: Splitter = new Splitter({
        height: '308px',
        paneSettings: [
            { size: '150px', min: '20%' },
            { size: '100px', min: '20%' }
        ],
        orientation: 'Vertical'
    });
    verticalObj.appendTo('#vertical_splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="Horizontal_splitter">
      <div>
        <div class="content">
          <h3 class="h3">PARIS </h3>
          Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">CAMEMBERT </h3>
          The village in the Orne département of Normandy where
the famous French cheese is originated from.
        </div>
      </div>
      <div>
        <div id="vertical_splitter" class="vertical_splitter">
          <div>
            <div class="content">

```

```

        <h3 class="h3">GRENOBLE </h3>
        The capital city of the French Alps and a major
        scientific center surrounded by many ski resorts, host of the Winter
        Olympics in 1968.
    </div>
</div>
<div>
    <div class="content">
        <h3 class="h3">Australia </h3>
        Australia is a country and continent surrounded
        by the Indian and Pacific oceans
    </div>
</div>
</div>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add or remove pane

You can add the panes programmatic but it will makes you complex. For this, you can use the `addPane/removePane` methods to add and remove the panes dynamically in the splitter.

Add pane

You can add the panes dynamically in the splitter by passing [paneProperties](#) along with index to the [addPane](#) method

INDEX.TS

```
import { Splitter, PanePropertiesModel } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
    height: '200px',
    paneSettings: [
        { size: '190px' },
        { size: '190px' }
    ],
    width: '600px',
    created: onSplitterCreate
});
splitObj.appendTo('#splitter');
let paneDetails : PanePropertiesModel = {
    size: '190px',
    content: 'New Pane',
    min: '30px',
    max: '250px',
}
function onSplitterCreate() {
    document.getElementById('addpane').addEventListener('click', () => {
        splitObj.addPane(paneDetails, 1);
    });
}
```

```

    })
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="splitter">
      <div>
        <div class="content">
          Pane 1
        </div>
      </div>
      <div>
        <div class="content">
          Pane 2
        </div>
      </div>
      <button id="addpane" class="e-btn">Add Pane</button>
    </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Remove pane

You can remove the split panes dynamically by passing the pane index to [removePane](#) method.

INDEX.TS

```
import { Splitter } from '@syncfusion/ej2-layouts';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
let splitObj: Splitter = new Splitter({
    height: '200px',
    paneSettings: [
        { size: '200px' },
        { size: '200px' },
        { size: '200px' }
    ],
    width: '600px'
});
splitObj.appendTo('#splitter');
document.getElementById('removepane').addEventListener('click', () => {
    if
    (!isNullOrUndefined(document.querySelector('#splitter').querySelectorAll('.e-
-pane-horizontal')[1]))
    {
        splitObj.removePane(1);
    }
})
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Splitter </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Uploader Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the splitter-->
        <div id="splitter">
            <div>
                <div class="content">
                    Pane 1
                </div>
            </div>
        </div>
    </div>
```

```

        <div class="content">
            Pane 2
        </div>
    </div>
    <div>
        <div class="content">
            Pane 3
        </div>
    </div>
    <button id="removepane" class="e-btn">Remove Pane</button>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Resizable split panes](#)
- [Collapsible panes](#)
- [Define size to a panes](#)
- [Specify content to a panes](#)

Expand and collapse in EJ2 JavaScript Splitter control

Collapsible panes

The Splitter panes can be configured with built-in expand and collapse functionalities. By default, the collapsible behavior is disabled. Enable the [collapsible](#) behavior in the paneSettings property to show or hide the expand or collapse icons in the panes. You can dynamically expand and collapse the panes by the corresponding icons.

The following code shows how to enable collapsible behavior.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
    height: '250px',
    paneSettings: [
        { collapsible: true },
        { collapsible: true },
        { collapsible: true }
    ],
    separatorSize: 2,
    width: '100%'
});
splitObj.appendTo('#splitter');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter Expand/Collapse </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
  <div class="default">
    <div id="splitter">
      <div>
        <div class="content">
          <h3 class="h3">PARIS </h3>
          Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">CAMEMBERT </h3>
          The village in the Orne department of Normandy where the
famous French cheese is originated from.
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">GRENOBLE </h3>
          The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
        </div>
      </div>
    </div>
  </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Programmatically control the expand and collapse action

The Splitter provides public method to control the expand and collapse behavior programmatically using the [expand](#) and [collapse](#) methods. Refer to the following code example.

INDEX.TS

```
import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
    height: '230px',
    paneSettings: [
        { collapsible: true },
        { collapsible: true },
        { collapsible: true }
    ],
    width: '100%'
});
splitObj.appendTo('#splitter');
document.getElementById('expand').onclick = (): void => {
    splitObj.expand(0);
}
document.getElementById('collapse').onclick = (): void => {
    splitObj.collapse(0);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Splitter </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Uploader Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <!--element which is going to render the splitter-->
    <div class="default">
        <div id="splitter">
            <div>
                <div class="content">
```

```

        <h3 class="h3">PARIS </h3>
        Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
    </div>
</div>
<div>
    <div class="content">
        <h3 class="h3">CAMEMBERT </h3>
        The village in the Orne department of Normandy where the
famous French cheese is originated from.
    </div>
</div>
<div>
    <div class="content">
        <h3 class="h3">GRENOBLE </h3>
        The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
    </div>
</div>
</div>
<button id="expand" class="e-control e-btn e-lib">expand</button>
<button id="collapse" class="e-control e-btn e-
lib">collapse</button>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Specify initial state to panes

You can render specific panes with collapsed state on page load. Specify a Boolean value to the [collapsed](#) property to control this behavior. The following code explains how to collapse panes on rendering (the second panes renders with collapsed state).

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
    height: '250px',
    paneSettings: [
        { collapsible: true },
        { collapsible: true, collapsed: true },
        { collapsible: true }
    ],
    width: '100%'
});
splitObj.appendTo('#splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
  <div class="default">
    <div id="splitter">
      <div>
        <div class="content">
          <h3 class="h3">PARIS </h3>
          Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">CAMEMBERT </h3>
          The village in the Orne department of Normandy where the
famous French cheese is originated from.
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">GRENOBLE </h3>
          The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
        </div>
      </div>
    </div>
  </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Resizable split panes](#)

Resizing in EJ2 JavaScript Splitter control

By default, resizing will be enable for split panes. Resizing gripper element will be add to the separator to makes the resize easy.

Horizontal splitter will allows to resize in horizontal directions.

Vertical splitter will allows to resize in vertical directions.

While resizing, previous and next panes will be adjust its dimensions automatically.

Min and Max validation

Splitter allows you to set the [minimum](#) and [maximum](#) sizes for each pane. Resizing will not be occur over the minimum and maximum values.

INDEX.TS

```
import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
  height: '200px',
  paneSettings: [
    { size: '200px', min: '20%', max: '40%' },
    { size: '200px', min: '30%', max: '60%' },
    { size: '200px' }
  ],
  width: '600px'
});
splitObj.appendTo('#splitter');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```

<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="splitter">
      <div>
        <div class="content">
          <h3 class="h3">Grid </h3>
          The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">Schedule </h3>
          The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">Chart </h3>
          ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
        </div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent resizing

You can disable the resizing for the pane by setting `false` to the `resizable` API within `paneSettings`.

Splitter resizing will be enabled only when the target of the adjacent pane's `resizable` api should also be in `true` state.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
  height: '200px',
  paneSettings: [
    { size: '200px', min: '20%', max: '40%', resizable: false },
    { size: '200px', min: '30%', max: '60%' },
    { size: '200px' }
  ],
  width: '600px'
});

```

```
splitObj.appendTo('#splitter');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <!--element which is going to render the splitter-->
    <div id="splitter">
      <div>
        <div class="content">
          <h3 class="h3">Grid </h3>
          The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">Schedule </h3>
          The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
        </div>
      </div>
      <div>
        <div class="content">
          <h3 class="h3">Chart </h3>
          ASP.NET charts, a well-crafted easy-to-use charting
package, is used to add beautiful charts in web and mobile applications
        </div>
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Refresh content on resizing

While resizing the panes, you can refresh the pane contents by using either [resizeStart](#), [resizing](#) or [resizeStop](#) events.

Customize the resize grip and cursor

You can customize the resize gripper icon and cursor in css level.

INDEX.TS

```

import { Splitter } from '@syncfusion/ej2-layouts';
let splitObj: Splitter = new Splitter({
  height: '200px',
  paneSettings: [
    { size: '200px', min: '20%', max: '40%' },
    { size: '200px', min: '30%', max: '60%' },
    { size: '200px' }
  ],
  width: '600px'
});
splitObj.appendTo('#splitter');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Splitter </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Uploader Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
layouts/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <style>
    .default-splitter .e-splitter .e-split-bar .e-resize-handler:before {
      content: "\e934";
      font-family: 'e-icons';
      font-size: 11px;
      transform: rotate(90deg);
    }
    .e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-bar,
    .e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-
bar::after {
      cursor: e-resize;
    }
  </style>

```



```

    }
</style><script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="container default-splitter">
        <!--element which is going to render the splitter-->
        <div id="splitter">
            <div>
                <div class="content">
                    <h3 class="h3">PARIS </h3>
                    Paris, the city of lights and love - this short guide is
full of ideas for how to make the most of the romanticism...
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">CAMEMBERT </h3>
                    The village in the Orne département of Normandy where
the famous French cheese is originated from.
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">GRENOBLE </h3>
                    The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
                </div>
            </div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Collapsible panes](#)

Different layouts in EJ2 JavaScript Splitter control

By using splitter control, you can create the different layouts with multiple and nested panes.

Code editor style layout

Step 1:

Create the element with two child to render the outer splitter.

```

`
<div id="VerticalSplitter">
<div id="HorizontalSplitter">
</div>
<div>
<div class="content">
<h3 class="h3">Preview of sample</h3>
<div class="splitter-image">

</div>
</div>
</div>
</div>
</div>
`

```

```

`javascript
let VerticalSplitObj: Splitter = new Splitter({
height: '400px',
paneSettings: [
{ size: '53%', min: '30%' },
],
orientation: 'Vertical'
});
VerticalSplitObj.appendTo('#VerticalSplitter');
`

```

Step 2 :

Render the first pane of vertical splitter as a horizontal splitter.

```

<div id="VerticalSplitter">
<div id="HorizontalSplitter">
<div>
<div class="content">
<h3 class="h3">HTML</h3>

```

```

<div class="code-preview">
<<span>!DOCTYPE html></span>
<div><<span>html></span></div>
<div><<span>body></span></div>
<<span>div</span> id="custom-image">
<div style="margin-left: 5px"><<span>img</span>src="src/albert.png"></div>
<div><<span>div</span>></div>
<div><<span>/body></span></div>
<div><<span>/html></span></div>
</div>
</div>
</div>
<div>
<div class="content">
<h3 class="h3">CSS</h3>
<div class="code-preview">
<span>img {</span>
<div id="code-text">margin:<span>0 auto;</span></div>
<div id="code-text">display:<span>flex;</span></div>
<div id="code-text">height:<span>70px;</span></div>
<span> }</span>
</div>
</div>
</div>
<div>
<div class="content">
<h3 class="h3">JavaScript</h3>
<div class="code-preview">
<span>var</span> image = document.getElementById("custom-image");
<div>image.addEventListener("click", function() {</div>
<div style="padding-left: 20px;">// Code block for click action</div>
<span> }</span>
</div>

```

```

</div>
</div>
</div>
<div class="content">
<h3 class="h3">Preview of sample</h3>
<div class="splitter-image">

</div>
</div>
</div>
</div>
`
`javascript
let HorizontalSplitObj: Splitter = new Splitter({
height: '220px',
paneSettings: [
{ size: '29%', min: '23%' },
{ size: '20%', min: '15%' },
{ size: '35%', min: '35%' }
],
width: '100%'
});
HorizontalSplitObj.appendTo('#HorizontalSplitter');
`
`
.code-preview {
margin-top: 15px;
font-size: 12px;
}
.content {
padding: 12px;
}
.splitter-image {

```

```
margin: 0 auto;
display: flex;
height: 115px;
margin-top: 10px;
}
,
```

Once the above configurations has been completed, you will get the output like [this](#)

Outlook style layout

Step 1:

Create the element with three panes and place the elements within the pane to render **treeview**, **listview** and **RTE**.

,

```
<div id="splitter1">
<div>
<div class="content">
<div id="tree"></div>
</div>
</div>
<div>
<div class="content">
<div id="groupedList" tabIndex="1"></div>
</div>
</div>
<div>
<div class="content">
<div style="width: 100%; padding: 15px;">
```

To...	
Cc...	
Subject	

```
</div>
<div class="forum">
<div id="createpostholder">
```

```

<div id="defaultRTE"> </div>
<div id="buttonSection">
<button class="e-btn e-primary" id="send">Send</button>
<button class="e-btn" id="discard">Discard</button>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
,

```

Step 2 :

Place the template script to render the treeview.

```

`javascript
<script id="treeTemplate" type="text/x-template">
<div>
<div class="treeviewdiv">
<div style="float:left">
<span class="treeName">${name}</span>
</div>
${if(count)}
<div style="margin-right: 5px; float:right">
<span class="treeCount e-badge e-badge-primary">${count}</span>
</div>
${/if}
</div>
</div>
</div>
</script>
,

```

Step 3 :

Render the listed controls one by one.

```

`javascript
let splitObj1: Splitter = new Splitter({

```

```
height: '493px',
paneSettings: [
{ size: '28%', min: '27%' },
{ size: '33%', min: '23%' },
{ size: '37%', min: '30%' }
],
width: '100%'
});
splitObj1.appendTo('#splitter1');
let inputobj1: TextBox = new TextBox({
});
inputobj1.appendTo('#firstname');
let inputobj2: TextBox = new TextBox({
});
inputobj2.appendTo('#lastname');
let inputobj3: TextBox = new TextBox({
});
inputobj3.appendTo('#subject');
// Data source for TreeView component
let mailBox: { [key: string]: Object }[] = [
{ id: 1, name: 'Favorites', hasChild: true},
{ id: 2, pid: 1, name: 'Sales Reports', count: '4' },
{ id: 3, pid: 1, name: 'Sent Items'},
{ id: 4, pid: 1, name: 'Marketing Reports ', count: '6'},
{ id: 5, name: 'Andrew Fuller', hasChild: true, expanded: true },
{ id: 6, pid: 5, name: 'Inbox', selected: true , count: '20'},
{ id: 7, pid: 5, name: 'Drafts', count: '5'},
{ id: 15, pid: 5, name: 'Archive'},
{ id: 8, pid: 5, name: 'Deleted Items'},
{ id: 9, pid: 5, name: 'Sent Items'},
{ id: 10, pid: 5, name: 'Sales Reports' , count: '4'},
{ id: 11, pid: 5, name: 'Marketing Reports', count: '6' },
{ id: 12, pid: 5, name: 'Outbox'},
```

```

{ id: 13, pid: 5, name: 'Junk Email'},
{ id: 14, pid: 5, name: 'RSS Feed'},
{ id: 15, pid: 5, name: 'Trash' }
];

// Render the TreeView using template option
let treeObj: TreeView = new TreeView({
fields: { dataSource: mailBox, id: 'id', parentID: 'pid', text: 'name', hasChildren: 'hasChild' },
nodeTemplate: '#treeTemplate',
});
treeObj.appendTo('#tree');

// tslint:disable:max-line-length
//Define an array of JSON data
let dataSource: any = [
{ Name: 'Selma Tally', content: '<div><div class="status">Apology marketing email</div><div id="list-
message">Hello Ananya Singleton</div>', id: '1', order: 0 },
{ Name: 'Illa Russo', content: '<div><div class="status">Annual conference</div><div id="list-
message">Hi jeani Moresa</div></div>', id: '4', order: 0 },
{ Name: 'Camden Macmellon', content: '<div><div class="status">Reference request- Camden
hester</div><div id="list-message">Hello Kerry Best,</div></div>', order: 0 },
{ Name: 'Garth Owen', content: '<div><div class="status">Application for job Title</div><div id="list-
message">Hello Illa Russo</div></div>', id: '2', order: 0 },
{ Name: 'Ursula Patterson', content: '<div><div class="status">Programmaer Position
Applicant</div><div id="list-message">Hello Kerry best,</div></div>', id: '2', order: 0 }
];

// Initialize ListView component
let listObj: ListView = new ListView({
//Set defined data to dataSource property
dataSource: dataSource,
cssClass: 'e-list-template',
//Map the appropriate columns to fields property
fields: { text: 'Name', groupBy: 'order' },
//Set customized group-header template
groupTemplate: '<div class="e-list-wrapper"><span class="e-list-item-content"></span></div>',
//Set customized list template
template: '<div class="settings e-list-wrapper e-list-multi-line e-list-avatar">' +

```



```
'<span class="e-list-item-header">${Name}</span>' +
'<span class="e-list-content">${content}</span>' +
'</div>'
});
//Render initialized ListView component
listObj.appendTo('#groupedList');
let button1: Button = new Button({ isPrimary: true });
button1.appendTo('#rteSubmit');
let button2: Button = new Button();
button2.appendTo('#rteCancel');
let defaultRTE: RichTextEditor = new RichTextEditor({ height: '262px'});
defaultRTE.appendTo('#defaultRTE');
`
`
<style>
target {
margin: 20px auto;
max-width: 820px;
}
.e-treeview .e-list-text {
width: 100%;
}
groupedList.e-listview .e-list-group-item {
height: 0;
}
splitter1 .settings.e-list-wrapper.e-list-multi-line.e-list-avatar {
padding: 15px;
}
buttonSection {
padding: 7px;
}
</style>
`
```

Once the above configurations has been completed, you will get the output like [this](#).

See Also

- [Multiple panes in Splitter](#)

Style in EJ2 JavaScript Splitter control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the split bar

Use the following CSS to customize the split bar properties.

Horizontal split bar

```
`css

/ default split bar color /

.e-splitter .e-split-bar.e-split-bar-horizontal{
background: blue;
}

/ split bar color in hover and active state /

.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active {
background: green;
}
`
```

Vertical split bar

```
`css

/ default split bar color /

.e-splitter .e-split-bar.e-split-bar-vertical {
background: blue;
}

/ split bar color in hover and active state /

.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover,
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active {
background: green;
}
`
```

Customizing the split bar resize handle

Use the following CSS to customize the split bar resize handle.

Horizontal split bar resize handle``css`*/ default split bar resize handle color /*`.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {``color: rgba(20, 27, 233, 0.54);``}`*/ default split bar resize handle color in hover and active state /*`.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-resize-handler,``.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-resize-handler {``color: green;``}``,`*Vertical split bar resize handle*``css`*/ default split bar resize handle color /*`.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {``color: rgba(20, 27, 233, 0.54);``}`*/ default split bar resize handle color in hover and active state /*`.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-resize-handler,``.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-resize-handler {``color: green;``}``,`*Customizing the split bar arrows*

Use the following CSS to customize the split bar arrows.

Horizontal split bar resize arrows``css`*/ split bar arrows /*`.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::before, .e-``splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::before, .e-``splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::after, .e-``splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::after, .e-``splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::before, .e-``splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::before, .e-`

```
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::after, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::after {
background-color: green;
}
```

/ split bar arrows - circular border /

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right {
border-color: rgba(33, 227, 22, 0.5);
}
```

,

Vertical split bar resize arrows

`css

/ split bar arrows /

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::after {
```

```
background-color: green;
```

```
}
```

/ split bar arrows - circular border /

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down {
```

```
border-color: rgba(33, 227, 22, 0.5);
```

```
}
```

,

To hide the resize handle in Splitter

Use the following CSS to hide the resize handler in the split bar

Hide the horizontal split bar resize arrow

`css

```
.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {
```

```
display: none;
```

```
}
```

,

Hide the vertical split bar resize arrow

```
`css
.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {
display: none;
}
`
```

Accessibility in EJ2 JavaScript Splitter control

The Splitter component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Splitter component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

Keyboard interaction

You can use the following key shortcuts to access the splitter without interruptions:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Tab | Helps in focusing the splitter on the page and switching between the consecutive splitter bars. |

| Shift + Tab | Helps in focusing the previous splitter bar element on the splitter. |

| Right arrow | Helps in moving the active horizontal orientated splitter bar to its Right side. |

| Left arrow | Helps in moving the active horizontal orientated splitter bar to its Left side. |

| Up arrow | Helps in moving the active vertical orientated splitter bar to its Up side. |

| Down arrow | Helps in moving the active vertical orientated splitter bar to its Down side. |

| Enter | Helps to toggle between expand and collapse actions of the splitter bar when it is active. |

Ensuring accessibility

The Splitter component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Splitter component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Splitter component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Ej1 api migration in EJ2 JavaScript Splitter control

This article describes the API migration process of Splitter component from Essential JS 1 to Essential JS 2.

Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Adding custom class | **Property:** `cssClass`
 < /> \$("#splitter").ejSplitter({
 cssClass: "custom-class"
 });
 | **Property:** `cssClass`
 < /> var obj = new ej.layouts.Splitter({
 cssClass: "custom-class"
 });
 obj.appendTo("#ej2_splitter") |

| Adjusting Height | **Property:** *height* `

 $("#splitter").ejSplitter({
Property: height

 var obj = new ej.layouts.Splitter({`

| Adjusting Width | **Property:** *width* `

 $("#splitter").ejSplitter({
Property: width

 var obj = new ej.layouts.Splitter({`

| Orientation | **Property:** *orientation* `

 $("#splitter").ejSplitter({
Property: orientation

 var obj = new`

| Separator Size | Not Available | **Property:** *separatorSize* `

 var obj = new ej.layouts.Splitter({`

| Adding HTML attributes | **Property:** *htmlAttributes* `

 $("#splitter").ejSplitter({`

| Customize expand/collapse icons | **Property:** `
 expanderTemplate`

| Make control flexible for mobile view | **Property:** *isResponsive* `

 $("#splitter").ejSplitter({`

| Refresh the Splitter | **Method:** *refresh()* `

 $('#splitter').ejSplitter('refresh')
Method: refresh()

 var obj = new ej.layouts.Splitter()`

| Destroy the Control | **Method:** *destroy()* `

 $('#splitter').ejSplitter('destroy')
Method: destroy()

 var obj = new ej.layouts.Splitter()`

| Event triggers after the Splitter created successfully | **Event:** *create* `

 $("#splitter").ejSplitter({
Event: created`

| Event triggers when Splitter has been destroyed | **Event:** *destroy* `

 $("#splitter").ejSplitter({`

Accessibility and Localization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Keyboard Navigation | **Property:** *allowKeyboardNavigation* `

 $("#splitter").ejSplitter({`

| Right to Left | **Property:** *enableRTL* `

 $("#splitter").ejSplitter({
Property: enableRtl

 var obj = new`

Control State

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable/Disable the control | Not Available | **Property:** *enabled* `

 var obj = new
 ej.layouts.Splitter({
 enabled: true
 })
 obj.appendTo("#ej2_splitter")` |

State Maintenance

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Save the model value in local storage or cookies | Not Available | **Property:** *enablePersistence* `

 var obj = new
 ej.layouts.Splitter({
 enablePersistence: false
 })
 obj.appendTo("#ej2_splitter")` |

Pane Properties

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** *properties* `

 $("#splitter").ejSplitter({
 properties: []
 });
` | **Property:** *paneSettings* `

 var obj = new
 ej.layouts.Splitter({
 paneSettings: []
 })
 obj.appendTo("#ej2_splitter")` |

| Pane Content | Not Available | **Property:** *content* `

 var obj = new
 ej.layouts.Splitter({
 paneSettings: [{ content:
'First Pane Content']
 })
 obj.appendTo("#ej2_splitter")` |

| Change the size of the pane | **Property:** *paneSize* `

 $("#splitter").ejSplitter({
 properties: [{ paneSize: "30px"}]
 });
` | **Property:** *size* `

 var obj = new
 ej.layouts.Splitter({
 paneSettings: [{ size: '25%'}]
 })
 obj.appendTo("#ej2_splitter")` |

| Minimum pane size | **Property:** *minSize* `

 $("#splitter").ejSplitter({
 properties: [{ minSize: 30 }]
 });
` | **Property:** *min* `

 var obj = new
 ej.layouts.Splitter({
 paneSettings: [{ min: '60px'}]
 })
 obj.appendTo("#ej2_splitter")` |

| Maximum pane size | **Property:** *maxSize* `

 $("#splitter").ejSplitter({
 properties: [{maxSize: 30 }]
 });
` | **Property:** *max* `

 var obj = new
 ej.layouts.Splitter({
 paneSettings: [{ max: '60px'}]
 })
 obj.appendTo("#ej2_splitter")` |

| Enable/Disable the Pane Resizable behavior | **Property:** *resizable* `

 $("#splitter").ejSplitter({
 properties: [{ resizable: false }]
 });
` | **Property:** *resizable* `

 var obj = new
 ej.layouts.Splitter({
 paneSettings: [{ resizable:
 false }]
 })
 obj.appendTo("#ej2_splitter")` |

| Collapsible | **Property:** *collapsible* `

 $("#splitter").ejSplitter({
 properties: [{ collapsible: true }]
 });
` | **Property:** *collapsible* `

 var obj = new
 ej.layouts.Splitter({
 paneSettings: [{ collapsible:
 true }]
 })
 obj.appendTo("#ej2_splitter")` |

| Expandable | **Property:** *expandable* `

 $("#splitter").ejSplitter({
 properties: [{expandable: true }]
 });
` | Not Available |

| Collapsed | Not Available | **Property:** *collapsed*

 var obj = new

ej.layouts.Splitter({
 paneSettings: [{ collapsed:
 true }]
 })

obj.appendTo("#ej2_splitter") |

| Add Pane | **Method:** *addItem()*

 \$("#splitter").ejSplitter("addItem",
 "New Pane 0", {

 paneSize: 20, minSize: 20, maxSize:
 100}, 0); | **Method:** *addPane()*

 var obj =
new
 ej.layouts.Splitter();
 obj.appendTo("#ej2_splitter")
 obj.addPane({size: "25%", <br

 content: "Pane"}, 0) |

| Remove Pane | **Method:** *removeItem()*

 \$("#splitter").ejSplitter("removeItem",
 0)<br

 | **Method:** *removePane()*

 var obj = new
 ej.layouts.Splitter();

obj.appendTo("#ej2_splitter")
 obj.removePane(0); |

| Collapse Pane | **Method:** *collapse()*

 \$("#splitter").ejSplitter("collapse", 0)
 | **Method:**
collapse()

 var obj = new
 ej.layouts.Splitter();
 obj.appendTo("#ej2_splitter") <br

 obj.collapse(0); |

| Expand Pane | **Method:** *expand()*

 \$("#splitter").ejSplitter("expand", 0)
 | **Method:**
expand()

 var obj = new
 ej.layouts.Splitter();
 obj.appendTo("#ej2_splitter")<br

 obj.expand(0); |

| Event triggers when before panes get expanded/collapsed | **Event:** *beforeExpandCollapse*

\$("#splitter").ejSplitter({
 beforeExpandCollapse: function
 (args) {}
 });
 | **Event:**
beforeExpand

 var obj = new ej.layouts.Splitter({
 beforeExpand: function() {}
 })

 obj.appendTo('#ej2splitter')

 Event: *beforeCollapse*
 var obj = new
ej.layouts.Splitter({
 beforeCollapse: function() {}
 })
 obj.appendTo('#ej2splitter') |

| Event triggers when after panes get expanded/collapsed | **Event:** *expandCollapse*
 <br

 \$("#splitter").ejSplitter({
 beforeCollapse: function
 (args) {}
 });
 | **Event:**
expand

 var obj = new ej.layouts.Splitter({
 expand: function() {}
 })

obj.appendTo('#ej2splitter')

 Event: *collapse*
 var obj = new ej.layouts.Splitter({

collapse: function() {}
 })
 obj.appendTo('#ej2splitter') |

| Event triggers when Resizing the pane | **Event:** *resize*

 \$("#splitter").ejSplitter({

resize: function
 (args) {}
 });
 | **Event:** *resizing*

 var obj = new
ej.layouts.Splitter({
 resizing: function() {}
 })
 obj.appendTo('#ej2_splitter') |

| Event triggers when pane is started to resize | Not Available | **Event:** *resizeStart*

 var obj =
new ej.layouts.Splitter({
 resizeStart: function() {}
 })
 obj.appendTo('#ej2_splitter') |

| Event triggers when pane is stopped to resize | Not Available | **Event:** *resizeStop*

 var obj =
new ej.layouts.Splitter({
 resizeStop: function() {}
 })
 obj.appendTo('#ej2_splitter') |

| Event triggers when click template icon | **Event:** *clickOnExpander*

\$("#splitter").ejSplitter({
 clickOnExpander: function
 (args) {}
 });
 | Not Available
|

Animation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| EnableAnimation | **Property:** *enableAnimation*

 \$("#splitter").ejSplitter({

enableAnimation: true
 });
 | Not Available |

| AnimationSpeed | **Property:** *animationSpeed*

 \$("#splitter").ejSplitter({
 animationSpeed: 150
 });
 | Not Available |

Spreadsheet

Overview of the EJ2 JavaScript Spreadsheet control

The Spreadsheet is an user interactive control to organize and analyze data in tabular format with configuration options for customization. It will load data by importing an Excel/CSV file or from local and remote data sources such as JSON, RESTful services, OData services, and more. The populated data can be exported as Excel with xlsx, xls, CSV and PDF formats.

Key features

- [Data sources](#): Bind the Spreadsheet control with an array of objects or data from a web service using **DataManager**.
- [Virtualization](#): Provides the option to load large amount of data without performance degradation.
- [Selection](#): Provides the option to select a cell or range of cells.
- [Editing](#): Provides the option to dynamically edit a cell.
- [Formulas](#): Provides built-in calculation library with pre-defined formulas and named range support.
- [Clipboard](#): Provides the option to perform clipboard operations.
- [Cell formatting](#): Provides the option to customize the appearance of cells.
- [Number formatting](#): Provides the option to format the cell value.
- [Open](#): Provides the option to open Excel and CSV files in Spreadsheet.
- [Save](#): Provides the option to save Spreadsheet data as Excel, CSV, and PDF documents.
- [Sorting](#): Helps you to arrange the data to particular order in a selected range of cells.
- [Filtering](#): Helps you to view specific rows in the Spreadsheet by hiding the other rows.
- [Undo Redo](#): Provides the option to perform undo redo operations in Spreadsheet.
- [Collaborative editing](#): Provides the option for real time changes across multiple users in the Spreadsheet.
- [Hyperlink](#): Provides the option to navigate to web link or cell reference within the sheet or to other sheet in Spreadsheet.
- [Resize](#): Allows you to change the row height and column width. Auto fit the rows and columns based on its content.
- [Wrap text](#): Provides the option to display the large content as multiple lines in a single cell.
- [Data validation](#): Provides the option to validate edited values based on data validation rules defined for a cell or range of cells.
- [Find and replace](#): Provides the option to find the data and replace it across all sheets in Spreadsheet.
- [Protect sheet](#): Provides the option to restrict user actions like cell editing, row and column insertion, deletion, and resizing.
- [Borders](#): Provides the option to customize cell gridlines such as color and its style for enhanced UI.
- [Show/hide](#): Provides the option to show/hide rows, columns and sheets.
- [Insert/delete](#): Provides the option to insert/delete rows, columns and sheets.
- [Merge cells](#): Provides the option to combine two or more cells located in the same row or column into a single cell.

- [Conditional formatting](#): Provides the option to format a cell or range of cells based on conditions applied.
- [Autofill](#): Provides the option to fill or copy a series or pattern of values and formats into adjacent cells in any direction.
- [Clear](#): Provides the option to clear the content, formats, and hyperlinks applied to a cell or range of cells in a Spreadsheet.
- [Aggregates](#): Provides the option to check the sum, average, count, and more for the selected cells or range in the sheet.
- [Picture](#): Allows you to view, insert, and modify a picture in a Spreadsheet with customizing options.
- [Chart](#): Transforms your Spreadsheet data to an intuitive overview for better understanding and to make smart business decisions.
- [Freeze panes](#): Allows you to keep the specified rows and columns always visible at the top and left side of the sheet while scrolling through the sheet.
- [Password protection](#): Allows you to protect the workbook with a password.
- [Multi-line editing](#): Allows you to insert a line break between paragraphs of the text within a cell in a Spreadsheet.
- [Calculate range selection](#): Helps you to select a range or multiple ranges when editing a formula in a cell.
- [Right-to-left \(RTL\)](#): Aligns content in the Spreadsheet control from right to left.
- [Templates](#): Templates can be used to create custom user experiences in the Spreadsheet.
- [Globalization](#): Personalize the Spreadsheet control with different languages, as well as culture-specific number, date, and time formatting.
- [Accessibility](#): Provides with built-in accessibility support which helps to access all the Spreadsheet control features through the keyboard, screen readers, or other assistive technology devices.

Getting started in EJ2 JavaScript Spreadsheet control

This section explains the steps to create a simple Essential JS 2 Spreadsheet control in a JavaScript application.

Dependencies

The following list of dependencies is required to use the Spreadsheet control in your application:

```
`js
|-- @syncfusion/ej2-spreadsheet
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-grids
`
```

Setup for local development

Refer to the following steps to setup your local environment.

Step 1: Create a root folder `myapp` for your application.

Step 2: Create `myapp/resources` folder to store local scripts and styles files.

Step 3: Create `myapp/index.js` and `myapp/index.html` files for initializing Essential JS 2 Spreadsheet control.

Adding Syncfusion resources

The Essential JS 2 Spreadsheet control can be initialized by using any of the following two ways:

- Using local script and style.
- Using CDN link for script and style.

Using local script and style

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the Spreadsheet and its dependencies scripts and style file into the resources folder.

Refer to the following code to find Spreadsheet's script and style file location.

Syntax:

Script: `(installed location)/Syncfusion/Essential Studio/JavaScript-EJ2/{RELEASEVERSION}/Web(Essential JS 2)/JavaScript/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js`

Styles: `(installed location)/Syncfusion/Essential Studio/JavaScript-EJ2/{RELEASEVERSION}/Web(Essential JS 2)/JavaScript/{PACKAGENAME}/styles/material.css`

Example:

Script: `C:/Program Files (x86)/Syncfusion/Essential Studio/JavaScript-EJ2/23.1.36/Web(Essential JS 2)/JavaScript/ej2-spreadsheet/dist/global/ej2-spreadsheet.min.js`

Styles: `C:/Program Files (x86)/Syncfusion/Essential Studio/JavaScript-EJ2/23.1.36/Web(Essential JS 2)/JavaScript/ej2-spreadsheet/styles/material.css`

After copying the files, then you can refer the Spreadsheet's scripts and styles into the `index.html` file. The following html code example shows the minimal dependency of Spreadsheet.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Spreadsheet</title>
<!-- Essential JS 2 Spreadsheet's dependents material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
```

```
<link href="resources/inputs/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/splitbuttons/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/lists/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/navigations/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/dropdowns/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/grids/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Spreadsheet's material theme -->
<link href="resources/spreadsheet/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Spreadsheet's dependents script -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-splitbuttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-excel-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-pdf-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calenders.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-grids.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-charts.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Spreadsheet global script -->
<script src="resources/scripts/ej2-spreadsheet.min.js" type="text/javascript"></script>
</head>
<body>
```

```
</body>
</html>
`
```

Using CDN link for script and style

Using CDN link, you can directly refer the Spreadsheet control's script and style into the `index.html`.

Refer to the Spreadsheet's CDN links as follows.

Syntax:

Script: `https://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `https://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-spreadsheet/dist/global/ej2-spreadsheet.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-spreadsheet/styles/material.css>

The following HTML code example shows the minimal dependency of Spreadsheet.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Spreadsheet</title>
<!-- Essential JS 2 Spreadsheet's dependents material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
```

```
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Spreadsheet material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-spreadsheet/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Spreadsheet's dependents script -->
<script src=http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-splitbuttons/dist/global/ej2-splitbuttons.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-excel-export/dist/global/ej2-excel-export.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-pdf-export/dist/global/ej2-pdf-export.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-calendars/dist/global/ej2-calendars.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-compression/dist/global/ej2-compression.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-file-utils/dist/global/ej2-file-utils.min.js
type="text/javascript"></script>
```

```
<script src=http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-svg-base/dist/global/ej2-svg-base.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-charts/dist/global/ej2-charts.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-spreadsheet/dist/global/ej2-spreadsheet.min.js
type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Add Spreadsheet control

Now, you can start adding Spreadsheet control in the application. For getting started, add a `div` element for Spreadsheet control in `index.html`. Then, refer the `index.js` file into the `index.html` file.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Spreadsheet</title>
<!-- Essential JS 2 Spreadsheet's dependents material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
```



```
<link href="http://cdn.syncfusion.com/ej2/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Spreadsheet material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-spreadsheet/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Spreadsheet's dependents script -->
<script src=http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-splitbuttons/dist/global/ej2-splitbuttons.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-inputs/dist/global/ej2-inputs.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-lists/dist/global/ej2-lists.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-dropdowns/dist/global/ej2-dropdowns.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-excel-export/dist/global/ej2-excel-export.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-pdf-export/dist/global/ej2-pdf-export.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-calendars/dist/global/ej2-calendars.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-compression/dist/global/ej2-compression.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-file-utils/dist/global/ej2-file-utils.min.js
```

```

type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-svg-base/dist/global/ej2-svg-base.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-charts/dist/global/ej2-charts.min.js
type="text/javascript"></script>
<script src=http://cdn.syncfusion.com/ej2/ej2-spreadsheet/dist/global/ej2-spreadsheet.min.js
type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for Spreadsheet -->
<div id="element"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Place the following Spreadsheet code in the `index.js`.

```

`javascript
//Initialize the Spreadsheet control
var spreadsheet = new ej.spreadsheet.Spreadsheet();
//Render the initialized Spreadsheet
spreadsheet.appendTo('#element');
`

```

[Run the application](#)

Now, run the `index.html` in web browser, it will render the Essential JS 2 Spreadsheet control.

Output will be displayed as follows.

INDEX.JS

```

// Initialize the Spreadsheet componenet.
var spreadsheet = new ej.spreadsheet.Spreadsheet({});
// Render initialized Spreadsheet.
spreadsheet.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Spreadsheet</title>

```

```

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript Spreadsheet Control">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Spreadsheet example](#) to know how to present and manipulate data, including editing, formulas, formatting, importing, and exporting.

See Also

- [Data Binding](#)
- [Open and Save](#)

Data binding in EJ2 JavaScript Spreadsheet control

The Spreadsheet uses [DataManager](#), which supports both RESTful JSON data services and local JavaScript object array binding to a range. The `dataSource` property can be assigned either with the instance of [DataManager](#) or JavaScript object array collection.

To bind data to a cell, use `cell data binding` support.

Local data

To bind local data to the Spreadsheet, you can assign a JavaScript object array to the `dataSource` property.

Refer to the following code example for local data binding.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [{
    ranges: [{ dataSource: data }],
    columns: [{ width: 80 }, { width: 80 }, { width: 80 },
              { width: 160 }, { width: 100 }, { width: 150 }]
  }],
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The local data source can also be provided as an instance of the [DataManager](#). By default, [DataManager](#) uses [JsonAdaptor](#) for local data-binding.

Remote data

To bind remote data to the Spreadsheet control, assign service data as an instance of [DataManager](#) to the `dataSource` property. To interact with remote data source, provide the service endpoint `url`.

Refer to the following code example for remote data binding.

INDEX.TS

```

import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
class CustomAdaptor extends ODataAdaptor {
    public processResponse(): Object {
        let result: Object[] = [];
        let original: { result: Object[], count: number } =
super.processResponse.apply(this, arguments);
        original.result.forEach((item: { SNo: number }, idx: number) => {
            result[idx] = {};

```

```

        Object.keys(item).forEach((key: string) => {
            if (['OrderID', 'CustomerID', 'ShipName', 'ShipCity',
'ShipCountry'].indexOf(key) > -1) {
                result[idx][key] = item[key];
            }
        });
    });
    return { result: result, count: original.count };
}
}
//Initialize DataManager.
let data: DataManager = new DataManager({
    //Remote service url.
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    adaptor: new CustomAdaptor,
    crossDomain: true
});
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [
        {
            rows: [{
                cells: [{ value: 'Order ID' }, { value: 'Customer Name' }, {
value: 'Ship Name' },
                { value: 'Ship City' }, { value: 'Ship Country' }]
            }],
            ranges: [{ dataSource: data, showFieldAsHeader: false,
startCell: 'A2' }],
            columns: [
                { width: 80 }, { width: 100 }, { width: 82 },
                { width: 160 }, { width: 110 }, { width: 130 }
            ]
        }
    ]
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="JavaScript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/core-js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

By default, **DataManager** uses **ODataAdaptor** for remote data-binding.

Binding with OData services

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the **DataManager**. Refer to the following code example for remote Data binding using OData service.

INDEX.TS

```

import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
//Initialize DataManager.
let data: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/Orders',
  adaptor: new ODataAdaptor(),
  crossDomain: true
});
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [

```

```

    {
      name: 'Order details',
      ranges: [{ dataSource: data }],
      columns: [
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 280 },
        { width: 180 },
        { width: 80 },
        { width: 180 },
        { width: 180 },
      ]
    }
  ],
  created: (): void => {
    //Applies cell and number formatting to specified range of the active
    sheet.
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center',
    verticalAlign: 'middle' },
      'A1:K1');
  }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="JavaScript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Web API

You can use WebApiAdaptor to bind spreadsheet with Web API created using OData endpoint.

INDEX.TS

```

import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
//Initialize DataManager.
let data: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/Orders',
  crossDomain: true,
  adaptor: new WebApiAdaptor()
});
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [
    {
      name: 'Order details',
      ranges: [{ dataSource: data }],
      columns: [
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 80 },
        { width: 280 },
        { width: 180 },
      ],
    },
  ],
});

```

```

        { width: 80 },
        { width: 180 },
        { width: 180 },
    ]
}
],
created: (): void => {
    //Applies cell and number formatting to specified range of the active
    sheet.
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center',
    verticalAlign: 'middle' },
    'A1:K1');
}
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="JavaScript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/core-
js/2.4.1/shim.min.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Cell data binding

The Spreadsheet control can bind the data to individual cell in a sheet . To achieve this you can use the **value** property.

Refer to the following code example for cell data binding.

INDEX.TS

```

import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [
    {
      name: 'Monthly Budget',
      selectedRange: 'D13',
      rows: [
        {
          cells: [
            { value: 'Category', style: { fontWeight: 'bold',
textAlign: 'center' } },
            { value: 'Planned cost', style: { fontWeight:
'bold', textAlign: 'center' } },
            { value: 'Actual cost', style: { fontWeight: 'bold',
textAlign: 'center' } },
            { value: 'Difference', style: { fontWeight: 'bold',
textAlign: 'center' } }
          ]
        },
        {
          cells: [
            { value: 'Food' },
            { value: '$7000' },
            { value: '$8120' },
            { formula: '=B2-C2', format: '$#,##0.00' }
          ]
        }
      ]
    },
    {
      cells: [

```

```

        { value: 'Loan' },
        { value: '$1500' },
        { value: '$1500' },
        { formula: '=B3-C3', format: '$#,##0.00' }
    ]
},
{
    cells: [
        { value: 'Medical' },
        { value: '$300' },
        { value: '$0' },
        { formula: '=B4-C4', format: '$#,##0.00' }
    ]
},
{
    index: 5,
    cells: [
        { index: 2, value: 'Total Difference:', style: {
fontWeight: 'bold', textAlign: 'right' } },
        { formula: '=D2+D4', format: '$#,##0.00', style: {
fontWeight: 'bold' } }
    ]
}
],
columns: [
    { width: 110 }, { width: 115 }, { width: 110 }, { width: 100
}
]
}
]);
//Render the initialized SpreadSheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The cell data binding also supports formula, style, number format, and more.

Dynamic data binding and Datasource change event

You can dynamically change the datasource of the spreadsheet by changing the `dataSource` property of the `range` object of the `sheet`. The `dataSourceChanged` event handler will be triggered when editing, inserting, and deleting a row in the datasource range. This event will be triggered with a parameter named `action` which indicates the `edit`, `add` and `delete` actions for the respective ones.

The following table defines the arguments of the `dataSourceChanged` event.

Property	Type	Description
action	string	Indicates the type of action such as <code>edit</code> , <code>add</code> , and <code>delete</code> performed in the datasource range.
data	object[]	Modified data for <code>edit</code> action; New data for <code>add</code> action; Deleted data for <code>delete</code> action.

| rangeIndex | number | Specifies the range index of the datasource. |

| sheetIndex | number | Specifies the sheet index of the datasource. |

For **add** action, the value for all the fields will be **null** in the data. In the case that you do not want the primary key field to be null which needs to be updated in the backend service, you can use **edit** action after updating the primary key field to update in the backend service.

For inserting a row at the end of the datasource range, you should insert a row below at the end of the range to trigger the **dataSourceChanged** event with action **add**.

INDEX.TS

```
import { Spreadsheet, DataSourceChangedEventArgs } from '@syncfusion/ej2-spreadsheet';
import { data, itemData } from './datasource.ts';
let spreadsheet: Spreadsheet = new Spreadsheet({
    showRibbon: false,
    showFormulaBar: false,
    sheets: [
        {
            ranges: [{ dataSource: data }],
            columns: [
                { width: 90 }, { width: 100 }, { width: 96 },
                { width: 120 }, { width: 130 }, { width: 120 }
            ]
        }
    ],
    dataSourceChanged: (args: DataSourceChangedEventArgs) => {
        appendElement("Data source changed with" + "<b>&#160;" +
args.action + "</b> action<hr>");
    }
});
spreadsheet.appendTo('#spreadsheet');
document.getElementById('changeDataBtn').addEventListener('click', () => {
    spreadsheet.sheets[0].ranges[0].dataSource = itemData;
});
document.getElementById('clearBtn').addEventListener('click', () => {
    document.getElementById('EventLog').innerHTML = "";
});
function appendElement(html: string): void {
    let span: HTMLElement = document.createElement("span");
    span.innerHTML = html;
    let log: HTMLElement = document.getElementById('EventLog');
    log.insertBefore(span, log.firstChild);
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
<div id="container">
  <div>
    <div>
      <button id="changeDataBtn" class="e-btn">Change
Datasource</button>
      <div id="spreadsheet"></div>
    </div>
    <div>
      <h4><b>Event Trace</b></h4>
      <div id="evt">
        <div style="height:173px;overflow: auto;min-
width: 250px;">
          <span id="EventLog" style="word-
break: normal;"></span>
        </div>
        <button id="clearBtn" class="e-
btn">Clear</button>
      </div>
    </div>
  </div>
</div><script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)
- [Collaborative Editing](#)

Open save in EJ2 JavaScript Spreadsheet control

The native data format for Spreadsheet is **JSON**. When you open an excel file, it needs to be read and converted to client side Spreadsheet model. The converted client side Spreadsheet model is sent as JSON which is used to render Spreadsheet. Similarly, when you save the Spreadsheet, the client Spreadsheet model is sent to the server as JSON for processing and saved as Excel file formats. [Server configuration](#) is used for this process.

Open

The Spreadsheet control opens an Excel document with its data, style, format, and more. To enable this feature, set [allowOpen](#) as **true** and assign service url to the [openUrl](#) property.

User Interface:

In user interface you can open an Excel document by clicking **File > Open** menu item in ribbon.

The following sample shows the **Open** option by using the [openUrl](#) property in the Spreadsheet control. You can also use the [beforeOpen](#) event to trigger before opening an Excel file.

INDEX.TS

```

import { Spreadsheet, BeforeOpenEventArgs } from '@syncfusion/ej2-spreadsheet';
//Initialize the Spreadsheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
    allowOpen: true,
    openUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/open',
    beforeOpen: (args: BeforeOpenEventArgs) => {
        // your code snippets here
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link rel="shortcut icon" href="resources/favicon.ico">
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Please find the below table for the beforeOpen event arguments.

Parameter	Type	Description
-----------	------	-------------

| ---- | ---- | ---- |

| file | FileList or string or File | To get the file stream. **FileList** - contains length and item index.

File - specifies the file lastModified and file name. |

| cancel | boolean | To prevent the open operation. |

| requestData | object | To provide the Form data. |

* Use **Ctrl + O** keyboard shortcut to open Excel documents.

* The default value of the [allowOpen](#) property is **true**. For demonstration purpose, we have showcased the [allowOpen](#) property in previous code snippet.

Open an external URL excel file while initial load

You can achieve to access the remote excel file by using the [created](#) event. In this event you can fetch the excel file and convert it to a blob. Convert this blob to a file and [open](#) this file by using Spreadsheet component open method.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
//Initialize the Spreadsheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
  openUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/open',
  saveUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/save',
  created: (): void => {

    fetch("https://cdn.syncfusion.com/scripts/spreadsheet/Sample.xlsx") // fetch
    the remote url
      .then((response) => {
        response.blob().then((fileBlob) => { // convert the
        excel file to blob
          let file = new File([fileBlob], "Sample.xlsx");
          //convert the blob into file
          spreadsheet.open({ file: file }); // open the file into
          Spreadsheet
        })
      })
  }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
  scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To add custom header during open

You can add your own custom header to the open action in the Spreadsheet. For processing the data, it has to be sent from server to client side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the [beforeOpen](#) event, the custom header can be added to the request during open action.

INDEX.TS

```
import { BeforeOpenEventArgs, Spreadsheet } from '@syncfusion/ej2-spreadsheet';
//Initialize the Spreadsheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
    openUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/open',
    saveUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/save',
    beforeOpen: (args: BeforeOpenEventArgs) => {
        args.requestData['headers'] = {
            Authorization: 'YOUR TEXT',
        }
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Open excel file into a read-only mode

You can open excel file into a read-only mode by using the [openComplete](#) event. In this event, you must protect all the sheets and lock its used range cells by using [protectSheet](#) and [lockCells](#) methods.

INDEX.TS

```

import {
    Spreadsheet,
    ProtectSettingsModel,
    SheetModel,
    getRangeAddress,
} from '@syncfusion/ej2-spreadsheet';
//Initialize Spreadsheet component
let spreadsheet: Spreadsheet = new Spreadsheet({
    openUrl:
        'https://services.syncfusion.com/js/production/api/spreadsheet/open',
    saveUrl:
        'https://services.syncfusion.com/js/production/api/spreadsheet/save',
    openComplete: (): void => {
        let sheets: SheetModel[] = spreadsheet.sheets;
        for (let index: number = 0; index < sheets.length; index++) {
            let name: string = spreadsheet.sheets[index].name;
            let protectSetting: ProtectSettingsModel = {
                selectCells: true,
                formatCells: false,
            };
            //To protect the sheet using sheet name
            spreadsheet.protectSheet(name, protectSetting);
            let address: string = getRangeAddress([
                0,
                0,
                sheets[index].usedRange.rowIndex,
                sheets[index].usedRange.colIndex,
            ]);
            //To lock the used range cells
            spreadsheet.lockCells(name + '!' + address, true);
        }
    },
},

```

```
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

External workbook confirmation dialog

When you open an excel file that contains external workbook references, you will see a confirmation dialog. This dialog allows you to either continue with the file opening or cancel the operation. This confirmation dialog will appear only if you set the `AllowExternalWorkbook` property value to **false** during the open request, as shown below. This prevents the spreadsheet from displaying inconsistent data.

```
`csharp
public IActionResult Open(IFormCollection openRequest)
{
    OpenRequest open = new OpenRequest();
    open.AllowExternalWorkbook = false;
    open.File = openRequest.Files[0];
    return Content(Workbook.Open(open));
}
```

This feature is only applicable when importing an Excel file and not when loading JSON data or binding cell data.

![External workbook confirmation dialog](./images/external-reference-dialog-alert%20.png)

Supported file formats

The following list of Excel file formats are supported in Spreadsheet:

- MS Excel (.xlsx)
- MS Excel 97-2003 (.xls)
- Comma Separated Values (.csv)
- Excel Macro-Enabled Workbook (.xlsm)
- Excel Binary Workbook(.xlsb)

Save

The Spreadsheet control saves its data, style, format, and more as Excel file document. To enable this feature, set `allowSave` as **true** and assign service url to the `saveUrl` property.

User Interface:

In user interface, you can save Spreadsheet data as Excel document by clicking **File > Save As** menu item in ribbon.

The following sample shows the **Save** option by using the `saveUrl` property in the Spreadsheet control. You can also use the `beforeSave` event to trigger before saving the Spreadsheet as an Excel file.

INDEX.TS

```

import { Spreadsheet, SheetModel, BeforeSaveEventArgs } from
 '@syncfusion/ej2-spreadsheet';
let sheet: SheetModel[] = [{
    rows: [{
        cells: [
            { index: 0, value: 'Order ID', style: { fontWeight:
'bold' } }},
            { value: 'Customer ID', style: { fontWeight: 'bold' } },
            { value: 'Employee ID', style: { fontWeight: 'bold' } },
            { value: 'Ship Name', style: { fontWeight: 'bold' } },
            { value: 'Ship City', style: { fontWeight: 'bold' } },
            { value: 'Ship Address', style: { fontWeight: 'bold' } }
        ]
    },
    {
        cells: [
            { value: '10248' },
            { value: 'VINET' },
            { value: '5' },
            { value: 'Vins et alcools Chevalier' },
            { value: 'Reims' },
            { value: '59 rue de lAbbaye' }
        ]
    },
    {
        cells: [
            { value: '10249' },
            { value: 'TOMSP' },
            { value: '6' },
            { value: 'Toms Spezialitäten' },
            { value: 'Münster' },
            { value: 'Luisenstr. 48' }
        ]
    },
    {
        cells: [
            { value: '10250' },
            { value: 'HANAR' },
            { value: '4' },
            { value: 'Hanari Carnes' },
            { value: 'Rio de Janeiro' },
            { value: 'Rua do Paço, 67' }
        ]
    },
    {
        cells: [
            { value: '10251' },
            { value: 'VICTE' },
            { value: '3' },
            { value: 'Victuailles en stock' },
            { value: 'Lyon' },
            { value: '2, rue du Commerce' }
        ]
    }
    ]},
    columns: [
        { width: 80 }, { width: 80 }, { width: 82 },
        { width: 160 }, { width: 110 }, { width: 130 }
    ]
}];

```



```

    ]
  }];
  //Initialize the Spreadsheet control
  let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheet,
    allowSave: true,
    saveUrl:
      'https://services.syncfusion.com/js/production/api/spreadsheet/save',
    beforeSave: (args: BeforeSaveEventArgs) => {
      // your code snippets here
    }
  });
  //Render the initialized Spreadsheet
  spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Please find the below table for the beforeSave event arguments.

Parameter	Type	Description
url	string	Specifies the save url.
fileName	string	Specifies the file name.
saveType	SaveType	Specifies the saveType like Xlsx, Xls, Csv and Pdf.
customParams	object	Passing the custom parameters from client to server while performing save operation.
isFullPost	boolean	It sends the form data from client to server, when set to true. It fetches the data from client to server and returns the data from server to client, when set to false.
needBlobData	boolean	You can get the blob data if set to true.
cancel	boolean	To prevent the save operations.

* Use **Ctrl + S** keyboard shortcut to save the Spreadsheet data as Excel file.

* The default value of [allowSave](#) property is **true**. For demonstration purpose, we have showcased the [allowSave](#) property in previous code snippet.

* Demo purpose only, we have used the online web service url link.

To send and receive custom params from client to server

Passing the custom parameters from client to server by using [beforeSave](#) event.

INDEX.TS

```

import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [{
    ranges: [{ dataSource: data }],
    columns: [{ width: 80 }, { width: 80 }, { width: 80 },
              { width: 160 }, { width: 100 }, { width: 150 }]
  }],

```

```

        saveUrl:
        'https://services.syncfusion.com/js/production/api/spreadsheet/save',
        beforeSave: (args: BeforeSaveEventArgs) => {
            args.customParams = { customParams: 'you can pass custom
params in server side'}; // you can pass the custom params
        }
    });
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">

```

```

        <div id="spreadsheet"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Server side code snippets:

```
`c#
```

```
public IActionResult Save(SaveSettings saveSettings, string customParams)
```

```
{
```

```
    Console.WriteLine(customParams); // you can get the custom params in controller side
```

```
    return Workbook.Save(saveSettings);
```

```
}
```

```
`
```

To add custom header during save

You can add your own custom header to the save action in the Spreadsheet. For processing the data, it has to be sent from client to server side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the [fileMenuItemSelect](#) event, the custom header can be added to the request during save action.

INDEX.TS

```

import { createElement } from '@syncfusion/ej2-base';
import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import {
    MenuSelectEventArgs,
} from '@syncfusion/ej2-spreadsheet';
let sheet: SheetModel[] = [{
    rows: [{
        cells: [
            { index: 0, value: 'Order ID', style: { fontWeight: 'bold' } },
            { value: 'Customer ID', style: { fontWeight: 'bold' } },
            { value: 'Employee ID', style: { fontWeight: 'bold' } },
            { value: 'Ship Name', style: { fontWeight: 'bold' } },
            { value: 'Ship City', style: { fontWeight: 'bold' } },
            { value: 'Ship Address', style: { fontWeight: 'bold' } }
        ]
    },
    {
        cells: [
            { value: '10248' },
            { value: 'VINET' },
            { value: '5' },
            { value: 'Vins et alcools Chevalier' },
            { value: 'Reims' },

```

```

    { value: '59 rue de l'Abbaye' }
  ]
},
{
  cells: [
    { value: '10249' },
    { value: 'TOMSP' },
    { value: '6' },
    { value: 'Toms Spezialitäten' },
    { value: 'Münster' },
    { value: 'Luisenstr. 48' }
  ]
},
{
  cells: [
    { value: '10250' },
    { value: 'HANAR' },
    { value: '4' },
    { value: 'Hanari Carnes' },
    { value: 'Rio de Janeiro' },
    { value: 'Rua do Paço, 67' }
  ]
},
{
  cells: [
    { value: '10251' },
    { value: 'VICTE' },
    { value: '3' },
    { value: 'Victuailles en stock' },
    { value: 'Lyon' },
    { value: '2, rue du Commerce' }
  ]
}
],
columns: [
  { width: 80 }, { width: 80 }, { width: 82 },
  { width: 160 }, { width: 110 }, { width: 130 }
]
}];
//Initialize the Spreadsheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheet,
  openUrl:
'https://services.syncfusion.com/js/production/api/spreadsheet/open',
  saveUrl:
'https://services.syncfusion.com/js/production/api/spreadsheet/save',
  fileMenuItemSelect: (args: MenuSelectEventArgs) => {
    if (args.item.text === 'Microsoft Excel') {
      args.cancel = true;
      spreadsheet.saveAsJson().then((response) => {
        var formData = new FormData();
        formData.append(
          'JSONData',
          JSON.stringify(response.jsonObject.Workbook)
        );
        formData.append('fileName', 'Sample');
        formData.append('saveType', 'Xlsx');
        fetch(

```

```

'https://services.syncfusion.com/js/production/api/spreadsheet/save',
    {
        method: 'POST',
        headers: { Authorization: 'YOUR TEXT' },
        body: formData,
    }
).then((response) => {
    response.blob().then((data) => {
        var anchor = createElement('a', {
            attrs: { download: 'Sample.xlsx' },
        });
        var url = URL.createObjectURL(data);
        anchor.href = url;
        document.body.appendChild(anchor);
        anchor.click();
        URL.revokeObjectURL(url);
        document.body.removeChild(anchor);
    });
});
});
});
},
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To change the PDF orientation

By default, the PDF document is created in **Portrait** orientation. You can change the orientation of the PDF document by using the `args.pdfLayoutSettings.orientation` argument settings in the [beforeSave](#) event.

The possible values are:

- **Portrait** - Used to display content in a vertical layout.
- **Landscape** - Used to display content in a horizontal layout.

INDEX.TS

```

import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{
        ranges: [{ dataSource: data }],
        columns: [{ width: 80 }, { width: 80 }, { width: 80 },
            { width: 160 }, { width: 100 }, { width: 150}]
    }],
    saveUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/save',
    beforeSave: (args: BeforeSaveEventArgs) => {
        args.pdfLayoutSettings.orientation = 'Landscape'; // You can
        change the orientation of the PDF document
    }
}

```

```

    }
  });
  spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";

```



```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Supported file formats

The following list of Excel file formats are supported in Spreadsheet:

- MS Excel (.xlsx)
- MS Excel 97-2003 (.xls)
- Comma Separated Values (.csv)
- Portable Document Format (.pdf)

Methods

To save the Spreadsheet document as an `xlsx`, `xls`, `csv`, or `pdf` file, by using [save](#) method should be called with the `url`, `fileName` and `saveType` as parameters. The following code example shows to save the spreadsheet file as an `xlsx`, `xls`, `csv`, or `pdf` in the button click event.

INDEX.TS

```

import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
//Initialize action items.
let items: ItemModel[] = [
  {
    text: "Save As xlsx"
  },
  {
    text: "Save As xls"
  },
  {
    text: "Save As csv"
  },
  {
    text: "Save As pdf"
  }
];
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton({
  items: items,
  cssClass: "e-round-corner",
  select: (args: MenuEventArgs) => {
    if (args.item.text === 'Save As xlsx')
      spreadsheet.save({url:
'https://services.syncfusion.com/js/production/api/spreadsheet/save',
fileName: "Sample", saveType: "Xlsx"});
    if (args.item.text === 'Save As xls')
      spreadsheet.save({url:
'https://services.syncfusion.com/js/production/api/spreadsheet/save',
fileName: "Sample", saveType: "Xls"});
    if (args.item.text === 'Save As csv')

```

```

        spreadsheet.save({url:
'https://services.syncfusion.com/js/production/api/spreadsheet/save',fileNam
e: "Sample", saveType: "Csv"});
        if (args.item.text === 'Save As pdf')
            spreadsheet.save({url:
'https://services.syncfusion.com/js/production/api/spreadsheet/save',fileNam
e: "Sample", saveType: "Pdf"});
    }
});
// Render initialized DropDownButton.
drpDownBtn.appendTo("#element");
let columns: ColumnModel[] = [{ width: 100 }, { width: 130 }, { width: 96},
    { width: 130 }, { width: 130 }, { width: 96},
    { width: 100 }, { width: 100 }, { width: 110}, { width: 100 }, { width:
130 }, { width: 150}]
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ ranges: [{ dataSource: data }], columns: columns }],
    allowSave: true
});
//Render the initialized Spreadsheet.
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

```

```

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <button id="element">Save</button>
    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Server Configuration

Import and export are processed in **server-side** using Spreadsheet server library. The following code snippets shows server configuration using **WebAPI** service,

```

`c#
[Route("api/[controller]")]
public class SpreadsheetController : Controller
{
    //To open excel file
    [AcceptVerbs("Post")]
    [HttpPost]
    [EnableCors("AllowAllOrigins")]
    [Route("Open")]
    public IActionResult Open(IFormCollection openRequest)
    {
        OpenRequest open = new OpenRequest();
        open.File = openRequest.Files[0];
        return Content(Workbook.Open(open));
    }
    //To save as excel file

```

```
[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("Save")]
public IActionResult Save([FromForm]SaveSettings saveSettings)
{
    return Workbook.Save(saveSettings);
}
}
```

Server Dependencies

Open and save helper functions are shipped in the Syncfusion.EJ2.Spreadsheet package, which is available in Essential Studio and [nuget.org](https://www.nuget.org). Following list of dependencies required for Spreadsheet open and save operations.

- Syncfusion.EJ2
- Syncfusion.EJ2.Spreadsheet
- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

And also refer [this](#) for more information.

See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

Worksheet in EJ2 JavaScript Spreadsheet control

Worksheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data.

Add sheet

You can dynamically add or insert a sheet by one of the following ways,

- Click the **Add Sheet** button in the sheet tab. This will add a new empty sheet next to current active sheet.
- Right-click on the sheet tab, and then select **Insert** option from the context menu to insert a new empty sheet before the current active sheet.
- Using [insertSheet](#) method, you can insert one or more sheets at your desired index.

The following code example shows the insert sheet operation in spreadsheet.

INDEX.TS

```

import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [{
    name: 'Price Details',
    ranges: [{ dataSource: data }],
    columns: [{ width: 150 }, { width: 110 }, { width: 110 }, { width: 85 },
{ width: 85 }, { width: 85 }, { width: 85 },
    { width: 85 }]
}];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: (): void => {
        // Applies style formatting to the active sheet before inserting a
        new sheet
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
        // inserting a new sheet with data at 1st index
        // You can also insert empty sheets by specifying the start and end
        sheet index instead of sheet model
        spreadsheet.insertSheet([{
            index: 1,
            name: 'Inserted Sheet',
            ranges: [{ dataSource: data }],
            columns: [{ width: 150 }, { width: 110 }, { width: 110 }, {
width: 85 }, { width: 85 }, { width: 85 }, { width: 85 },
            { width: 85 }]
        }]);
        // Applies style formatting for the inserted sheet
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Inserted Sheet!A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'Inserted
Sheet!D2:H11');
    },
    // Removed the unwanted support for this samples
    showRibbon: false, showFormulaBar: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Delete sheet

The Spreadsheet has support for removing an existing worksheet. You can dynamically delete the existing sheet by the following way,

- Right-click on the sheet tab, and then select **Delete** option from context menu.
- Using [delete](#) method to delete the sheets.

Rename sheet

You can dynamically rename an existing worksheet in the following way,

- Right-click on the sheet tab, and then select **Rename** option from the context menu.

Headers

By default, the row and column headers are visible in worksheets. You can dynamically show or hide worksheet headers by using one of the following ways,

- Switch to **View** tab, and then select **Hide Headers** option to hide both the row and column headers.
- Set **showHeaders** property in sheets as **true** or **false** to show or hide the headers at initial load. By default, the **showHeaders** property is enabled in each worksheet.

Gridlines

Gridlines act as a border like appearance of cells. They are used to distinguish cells on the worksheet. You can dynamically show or hide gridlines by using one of the following ways,

- Switch to **View** tab, and then select **Hide Gridlines** option to hide the gridlines in worksheet.
- Set **showGridLines** property in sheets as **true** or **false** to show or hide the gridlines at initial load. By default, the **showGridLines** property is enabled in each worksheet.

The following code example shows the headers and gridlines operation in spreadsheet.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 150 }, { width: 110 }, { width: 110 }, { width: 85 }, { width: 85 }, { width: 85 }, { width: 85 }, { width: 85 }];
let sheets: SheetModel[] = [{
  name: 'Price Details',
  ranges: [{ dataSource: data }],
  columns: columns,
  // Hiding the gridlines in 'Price Details' sheet
  showGridLines: false,
  // Hiding the headers in 'Price Details' sheet
  showHeaders: false
}];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,
  created: (): void => {
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' }, 'A1:H1');
    spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
    // The gridlines have been removed to set border for the range of cells
    spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A1:H11');
  },
  // Removed the unwanted support for this samples
  showFormulaBar: false, showSheetTabs: false
});
```

```
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```



```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Sheet visibility

Hiding a worksheet can help prevent unauthorized or accidental changes to your file.

There are three visibility state as like Microsoft Excel,

State	Description
Visible	You can see the worksheet once the component is loaded.
Hidden	This worksheet is not visible, but you can unhide by selecting the sheet from List All Sheets dropdown menu.
VeryHidden	This worksheet is not visible and cannot be unhidden. Changing the state property to Visible is the only way to view this sheet.

The following code example shows the three types of sheet visibility state.

INDEX.TS

```

import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
enableRipple(true);
let sheets: SheetModel[] = [{
  name: 'Visible Sheet',
  ranges: [{ dataSource: data }],
  columns: [{ width: 150 }, { width: 110 }, { width: 110 }, { width: 85 },
{ width: 85 }, { width: 85 }, { width: 85 },
{ width: 85 }],
  // By default, state is set as `visible`. We don't need to said it in
the sample
  state: 'Visible'
},
{
  name: 'Very Hidden Sheet',
  ranges: [{ dataSource: data }],
  columns: [{ width: 150 }, { width: 110 }, { width: 110 }, { width: 85 },
{ width: 85 }, { width: 85 }, { width: 85 },
{ width: 85 }],
  // Sets sheet state as `VeryHidden`. It can't be unhidden.
  state: 'VeryHidden'
},
{
  name: 'Hidden Sheet',
  ranges: [{ dataSource: data }],
  columns: [{ width: 150 }, { width: 110 }, { width: 110 }, { width: 85 },
{ width: 85 }, { width: 85 }, { width: 85 },
{ width: 85 }],
  // Sets sheet state as `Hidden`. It can be unhidden dynamically.
}

```

```

        state: 'Hidden'
    }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: (): void => {
        // Applies style formatting to active visible sheet
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
        'A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
        // Applies style formatting to active hidden sheet
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
        'Hidden Sheet!A1:H1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'Hidden
        Sheet!D2:H11');
    },
    openUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/open',
    saveUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/save',
    // Removed the unwanted support for this samples
    showFormulaBar: false, showRibbon: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Sheet protection](#)
- [Rows and columns](#)
- [Cell range](#)
- [Formatting](#)

Cell range in EJ2 JavaScript Spreadsheet control

A group of cells in a sheet is known as cell range.

Wrap text

Wrap text allows you to display large content as multiple lines in a single cell. By default, the wrap text support is enabled. Use the [allowWrap](#) property to enable or disable the wrap text support in spreadsheet.

Wrap text can be applied or removed to a cell or range of cells in the following ways,

- Using the `wrap` property in `cell`, you can enable or disable wrap text to a cell at initial load.
- Select or deselect wrap button from ribbon toolbar to apply or remove the wrap text to the selected range.
- Using the [wrap](#) method, you can apply or remove the wrap text once the component is loaded.

The following code example shows the wrap text functionality in spreadsheet.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
 '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ index: 1, width: 100 }, { width: 140 }, {
width: 90 }, { width: 150 }, { width: 120 }, { width: 90 }, { width: 180 }];
// Set wrap text to H2 to H5 cells through cell binding
let rows: RowModel[] = [{ height: 30 }, { cells: [{ index: 7, wrap: true } ]
}, { cells: [{ index: 7, wrap: true } ] }, { cells: [{ index: 7, wrap: true
} ] }, { cells: [{ index: 7, wrap: true } ] }];
let sheets: SheetModel[] = [{ name: 'Movie List', ranges: [{ dataSource:
data }], rows: rows, columns: columns }];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,
  created: (): void => {
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
    spreadsheet.cellFormat({ verticalAlign: 'middle' }, 'A1:H5');
    spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:B5');
    spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:D5');
    // To wrap the cells from E2 to E5 range
    spreadsheet.wrap('E2:E5');
    // To unwrap the H3 cell
    spreadsheet.wrap('H3', false);
  },
  // Removed the unwanted support for this samples
  showFormulaBar: false
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Wrap text

The following features have some limitations in wrap text:

- Sorting with wrap text applied data.
- Merge with wrap text.

Merge cells

Merge cells allows users to span two or more cells in the same row or column into a single cell. When cells with multiple values are merged, top-left most cell data will be the data for the merged cell. By default, the merge cells option is enabled. Use [allowMerge](#) property to enable or disable the merge cells option in spreadsheet.

You can merge the range of cells in the following ways,

- Set the `rowSpan` and `colSpan` property in `cell` to merge the number of cells at initial load.
- Select the range of cells and apply merge by selecting the desired option from ribbon toolbar.
- Use [merge](#) method to merge the range of cells, once the component is loaded.

The available merge options in spreadsheet are,

Type	Action
----- -----	
Merge All	Combines all the cells in a range in to a single cell (default).
Merge Horizontally	Combines cells in a range as row-wise.
Merge Vertically	Combines cells in a range as column-wise.
UnMerge	Splits the merged cells into multiple cells.

The following code example shows the merge cells operation in spreadsheet.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
 '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 90 }, { width: 150 }, { width: 100 },
 { width: 100 }, { width: 100 }, { width: 100 }, { width: 100 }, { width: 100 },
 { width: 100 }, { width: 100 }, { width: 120 }, { width: 120 }, { width:
 120 }, { width: 120 }, { width: 120 }, { width: 120 }, { width: 100 }, {
 width: 100 }, { width: 100 }];
let rows: RowModel[] = [{ height: 35 }, { height: 35, cells: [
  // Merging the 2nd cells of rows 2 and 3 through cell binding.
  { index: 1, rowSpan: 2 },
  // Merging the 2nd row's 3rd and 4th cells through cell binding.
  { colSpan: 2 },
  // Merging the 2nd row's 7th, 8th and 9th cells through cell binding.
  { index: 6, colSpan: 3 },
  // Merging the 2nd and 3rd rows 11th, 12th and 13th cells through cell
  binding.
  { index: 10, rowSpan: 2, colSpan: 3 },
  { index: 13, colSpan: 2 }, { index: 17, colSpan: 2 } ] },
 { height: 35, cells: [{ index: 3, colSpan: 3 }, { index: 6, colSpan: 4
 }, { index: 13, colSpan: 3 }, { index: 17, colSpan: 2 } ] },
 { height: 35, cells: [{ index: 2, colSpan: 3 }, { index: 5, colSpan: 2
 }, { index: 7, colSpan: 3 }, { index: 15, colSpan: 2 },
 { index: 17, colSpan: 2 } ] }, { height: 35, cells: [{ index: 2, colSpan:
 3 }, { index: 6, colSpan: 4 }, { index: 16, colSpan: 2 } ] },
 { height: 35, cells: [{ index: 2, colSpan: 4 }, { index: 7, colSpan: 3
 }, { index: 15, colSpan: 2 }, { index: 17, colSpan: 2 } ] }];
let sheets: SheetModel[] = [{ name: 'Merge Cells', ranges: [{ dataSource:
 data }], columns: columns, rows: rows }];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,
  created: (): void => {
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:S1');
    spreadsheet.numberFormat('h:mm AM/PM', 'C1:S1');
    spreadsheet.cellFormat({ verticalAlign: 'middle' }, 'A1:S11');
    // Merging the `K4:M4` cells using method
    spreadsheet.merge('K4:M4');
```

```

        // Merging the 5th and 6th row cells across 11th, 12th and 13th
        column
        spreadsheet.merge('K5:M6', 'Vertically');
        // Merging the 18th and 19th column cells across 2nd, 3rd and 4th
        row
        spreadsheet.merge('N4:O6', 'Horizontally');
    },
    // Removed the unwanted support for this samples
    showFormulaBar: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations of Merge

The following features have some limitations in Merge:

- Merge with filter.
- Merge with wrap text.

Data Validation

Data Validation is used to restrict the user from entering the invalid data. You can use the [allowDataValidation](#) property to enable or disable data validation.

* The default value for `allowDataValidation` property is `true`.

Apply Validation

You can apply data validation to restrict the type of data or the values that users enter into a cell.

You can apply data validation by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Data Validation item.
- Use the [addDataValidation\(\)](#) method programmatically.

Clear Validation

Clear validation feature is used to remove data validations from the specified ranges or the whole worksheet.

You can clear data validation rule by one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Validation item.
- Use the [removeDataValidation\(\)](#) method programmatically.

Highlight Invalid Data

Highlight invalid data feature is used to highlight the previously entered invalid values.

You can highlight an invalid data by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Highlight Invalid Data item.
- Use the [addInvalidHighlight\(\)](#) method programmatically.

Clear Highlighted Invalid Data

Clear highlight feature is used to remove the highlight from invalid cells.

You can clear the highlighted invalid data by using the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Highlight item.
- Use the [removeInvalidHighlight\(\)](#) method programmatically.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
let sheet: SheetModel[] = [
    {
        name: 'PriceDetails',
        rows: [
            {
                index: 0,
                cells: [{ index: 0, value: 'Seller Name',
style:{fontWeight : "bold", textAlign: "center"} },
                { index: 1, value: 'Customer Id', style:{fontWeight :
"bold", textAlign: "center"} },
                { index: 2, value: 'Customer Name', style:{fontWeight :
"bold", textAlign: "center"} },
                { index: 3, value: 'Product Name', style:{fontWeight :
"bold", textAlign: "center"} },
                { index: 4, value: 'Product Price', style:{fontWeight :
"bold", textAlign: "center"} },
                { index: 5, value: 'Sales Date', style:{fontWeight :
"bold", textAlign: "center"} },
                { index: 5, value: 'Billing Time', style:{fontWeight :
"bold", textAlign: "center"} },
                { index: 6, value: 'Total Price', style:{fontWeight :
"bold", textAlign: "center"} }]]
            },
            {
                index: 1,
                cells: [{ index: 0, value: 'John'},
                { index: 1, value: '1', validation: { type:
'WholeNumber', operator: 'NotEqualTo', value1: '1'}},
                { index: 2, value: 'Nash'},
                { index: 3, value: 'Digger', validation:{ type: 'List',
value1: 'Digger, Digger, Cherrypicker' }},
                { index: 4, value: '50000', validation:{ type: 'List',
value1: '50000,50000,45000' }},
                { index: 5, value: '04/11/2019'},
                { index: 6, value: '11:34:32 AM'},
                { index: 7, value: '1,45,000.00' }]]
            },
            {
                index: 2,
                cells: [{ index: 0, value: 'Mike'},
                { index: 1, value: '2', validation: { type:
'WholeNumber', operator: 'NotEqualTo', value1: '1'}},
                { index: 2, value: 'Jim'},
                { index: 3, value: 'Cherrypicker', validation:{ type:
'List', value1: 'Cherrypicker, JCB, Wheelbarrow' }},
```

```

        { index: 4, value: '45000', validation:{ type: 'List',
value1: '45000,90000,40' }},
        { index: 5, value: '04/11/2019'},
        { index: 6, value: '10:15:00 AM'},
        { index: 7, value: '1,40,040.00'}]
    },
    {
        index: 3,
        cells: [{ index: 0, value: 'shane' },
        { index: 1, value: '3', validation: { type:
'WholeNumber', operator: 'NotEqualTo', value1: '1'}},
        { index: 2, value: 'Sean'},
        { index: 3, value: 'Kango', validation:{ type: 'List',
value1: 'Kango, Ropes' }},
        { index: 4, value: '450', validation:{ type: 'List',
value1: '450, 95' }},
        { index: 5, value: '06/25/2019'},
        { index: 6, value: '01:30:11 PM'},
        { index: 7, value: '545.00'}]
    },
    {
        index: 4,
        cells: [{ index: 0, value: 'John' },
        { index: 1, value: '1', validation: { type:
'WholeNumber', operator: 'NotEqualTo', value1: '1'}},
        { index: 2, value: 'Nash'},
        { index: 3, value: 'JCB', validation:{ type: 'List',
value1: 'JCB, Ropes, scaffolding' }},
        { index: 4, value: '90000', validation:{ type: 'List',
value1: '90000, 95, 10000' }},
        { index: 5, value: '09/22/2019'},
        { index: 6, value: '12:30:02 PM'},
        { index: 7, value: '1,00,095.00' }]
    }
],
columns: [
    { width: 88, }, { width: 88 }, { width: 106 }, { width: 98 }, {
width: 88 }, { width: 86 }, { width: 107 }, { width: 81}
]
}
];
//Initialize the Spreadsheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheet,
    created: () => {
        //Add Data validation to range.
        spreadsheet.addDataValidation({ type: 'TextLength' , operator:
'LessThanOrEqualTo' , value1: '4' }, 'A2:A5');
        spreadsheet.addDataValidation({ type: 'WholeNumber', operator:
'NotEqualTo', value1: '1' }, 'B2:B5');
        spreadsheet.addDataValidation({ type: 'Date', operator:
'NotEqualTo', value1: '04/11/2019'}, 'F2:F5');
        spreadsheet.addDataValidation({ type: 'Time', operator:
'Between', value1: '10:00:00 AM', value2: '11:00:00 AM' }, 'G2:G5');
        spreadsheet.addDataValidation({ type: 'Decimal', operator:
'LessThan', value1: '100000.00'}, 'H2:H5');
        //Highlight Invalid Data.
    }
});

```

```

        spreadsheet.addInvalidHighlight('A1:H5');
    }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations of Data validation

The following features have some limitations in Data Validation:

- Entire row data validation.
- Insert row between the data validation.
- Copy/paste with data validation.
- Delete cells between data validation applied range.

Auto Fill

Auto Fill is used to fill the cells with data based on adjacent cells. It also follows a pattern from adjacent cells if available. There is no need to enter the repeated data manually. You can use `allowAutoFill` property to enable/disable the auto fill support. You can also use `showFillOptions` property to enable/disable the fill option and `fillType` property to change the default auto fill option which is available in `autoFillSettings`.

You can do this by one of the following ways,

- Using “AutoFillOptions” menu which is open, while drag and drop the cell using fill handle element.
- Use the `autoFill()` method programmatically.

The available parameters in `autoFill()` method are,

Parameter	Type	Description
fillRange	string	Specifies the fill range.
dataRange	string	Specifies the data range.
direction	AutoFillDirection	Specifies the direction("Up","Right","Down","Left") to be filled.
fillType	AutoFillType	Specifies the fill type("CopyCells","FillSeries","FillFormattingOnly","FillWithoutFormatting") for autofill action.

In Auto Fill we have following options,

- Copy Cells
- Fill Series
- Fill Formatting Only
- Fill Without Formatting

* The default auto fill option is “FillSeries” which can be referred from `fillType` property.

Copy Cells

To copy the selected cell content to the adjacent cells. You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Copy Cells” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “CopyCells” as fill type in `autoFill` method to fill the adjacent cells.

Fill Series

To fill the series of numbers, characters, or dates based on selected cell content to the adjacent cells with their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Series” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillSeries” as fill type in `autoFill` method to fill the adjacent cells.

Fill Formatting Only

To fill the cell style and number formatting based on the selected cell content to the adjacent cells without their content.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Formatting Only” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillFormattingOnly” as fill type in `autoFill` method to fill the adjacent cells.

Fill Without Formatting

To fill series of numbers, characters, or dates based on the selected cells to the adjacent cells without their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Without Formatting” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillWithoutFormatting” as fill type in `autoFill` method to fill the adjacent cells.

In the following sample, you can enable/disable the fill option on the button click event by using the `showFillOptions` property in `autoFillSettings`.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel, getFormatFromType }
  from '@syncfusion/ej2-spreadsheet';
import { Button } from '@syncfusion/ej2-buttons';
import { defaultData } from '../datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [
  {
    name: 'Price Details',
    ranges: [{ dataSource: defaultData, startCell: 'A1' }],
    rows: [
      {
        height: 30
      }
    ]
  }
];
```

```

    },
    rowCount: 200,
    columns: [
        { width: 130 }, { width: 100 }, { width: 100 }
    ]
}
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    autoFillSettings: { fillType: 'FillSeries', showFillOptions: true },
    created: function () {
        spreadsheet.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A1:H1');
        spreadsheet.autoFill('D4:D11', 'D2:D3', 'Down', 'CopyCells');
        spreadsheet.autoFill('E4:E11', 'E2:E3', 'Down', 'FillSeries');

        spreadsheet.autoFill('B4:B11', 'B2:B3', 'Down', 'FillFormattingOnly');

        spreadsheet.autoFill('C4:C11', 'C2:C3', 'Down', 'FillWithoutFormatting');
    }
});
spreadsheet.appendTo('#spreadsheet');
var button: Button = new Button();
button.appendTo('#showfillbtn');
button.element.onclick = (): void => {
    var spreadsheetObj =
document.getElementById("spreadsheet").ej2_instances[0];
    var showFillOptions = spreadsheetObj.autoFillSettings.showFillOptions;
    spreadsheetObj.autoFillSettings.showFillOptions = !showFillOptions; //To
change whether fill options need to be shown or not.
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <button id="showfillbtn">Change showFillOptions</button>
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Autofill

The following features have some limitations in Autofill:

- Flash Fill option in Autofill feature.
- Fill with Conditional Formatting applied cells.

Clear

Clear feature helps you to clear the cell contents (formulas and data), formats (including number formats, conditional formats, and borders) in a spreadsheet. When you apply clear all, both the contents and the formats will be cleared simultaneously.

Apply Clear Feature

You can apply clear feature by using one of the following ways,

- Select the clear icon in the Ribbon toolbar under the Home Tab.
- Using the [clear\(\)](#) method to clear the values.

Clear has the following types in the spreadsheet,

| Options | Uses |

|-----|-----|

| **Clear All** | Used to clear all contents, formats, and hyperlinks. |

| **Clear Formats** | Used to clear the formats (including number formats, conditional formats, and borders) in a cell. |

| **Clear Contents** | Used to clear the contents (formulas and data) in a cell. |

| **Clear Hyperlinks** | Used to clear the hyperlink in a cell. |

Methods

Clear the cell contents and formats in the Spreadsheet document by using the [clear](#) method. The [clear](#) method has **type** and **range** as parameters. The following code example shows how to clear the cell contents and formats in the button click event.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { orderData } from './datasource.ts';
//Initialize action items.
let items: ItemModel[] = [
    {
        text: "Clear All"
    },
    {
        text: "Clear Formats"
    },
    {
        text: "Clear Contents"
    },
    {
        text: "Clear Hyperlinks"
    }
];
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton({
    items: items,
    cssClass: "e-round-corner",
    select: (args: MenuEventArgs) => {
        if (args.item.text === 'Clear All')
            spreadsheet.clear({ type: 'Clear All', range: 'D1:D10' }); // Clear
            the content, formats and hyperlinks applied in the provided range.
        if (args.item.text === 'Clear Formats')
            spreadsheet.clear({ type: 'Clear Formats', range: 'B1:B10' }); //
            Clear the formats applied in the provided range
        if (args.item.text === 'Clear Contents')
            spreadsheet.clear({ type: 'Clear Contents', range: 'A1:A10' }); //
            Clear the content in the provided range
        if (args.item.text === 'Clear Hyperlinks')
            spreadsheet.clear({ type: 'Clear Hyperlinks', range: 'F2:F6' }); //
            Clear the hyperlinks applied in the provided range
    }
});
```



```

});
// Render initialized DropDownButton.
drpDownBtn.appendTo("#element");
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [{
    ranges: [{ dataSource: orderData }],
    columns: [{ width: 80 }, { width: 80 }, { width: 80 },
      { width: 160 }, { width: 100 }, {width: 150}]
  }],
  created: (): void => {
    spreadsheet.cellFormat({ fontWeight: 'bold', fontSize: '12pt',
'A1:E1' });
    spreadsheet.cellFormat({ color: '#10c469' }, 'B1:B10');
  },
}
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <button id="element">Clear</button>
    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Rows and columns](#)
- [Formatting](#)
- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)

Editing in EJ2 JavaScript Spreadsheet control

You can edit the contents of a cell directly in the cell or by typing in the formula bar. By default, the editing feature is enabled in the spreadsheet. Use the [allowEditing](#) property to enable or disable the editing feature.

Edit cell

You can start editing by one of the following ways,

- Double click a cell to start the edit mode.
- Press **F2** key to edit the active cell.
- Use formula bar to perform editing.
- Use **BACKSPACE** or **SPACE** key to clear the cell content and start the edit mode.
- Using the [startEdit](#) method.

Save cell

If the cell is in editable state, you can save the edited cell by one of the following ways,

- Perform mouse click on any other cell rather than the current editing cell.
- Press **Enter** or **Tab** keys to save the edited cell content.
- Using the [endEdit](#) method.

Cancel editing

To cancel the editing without saving the changes, you can use one of the following ways,

- Press **ESCAPE** key, this will remove the editable state and update the unchanged cell content.
- Using the [closeEdit](#) method.

The following sample shows how to prevent the editing and cell save. Here **E** column prevent the editing by using cancel argument as true in [cellEdit](#) event. In **D** column, prevent saving the edited changes by using cancel argument as true in [beforeCellSave](#) and use [closeEdit](#) method in spreadsheet.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
 '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 120 }, { width: 180 }, { width: 100
 }, { width: 120 }, { width: 120 }];
let rows: RowModel[] = [{
    index: 10, cells: [{ index: 3, value: 'Total Amount:', style: {
fontWeight: 'bold' } }, { formula: '=SUM(E2:E10)' }]
}];
let sheets: SheetModel[] = [{ ranges: [{ dataSource: data }], columns:
columns, selectedRange: 'E11', rows: rows }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:E1');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:A10');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'C2:C10');
        spreadsheet.numberFormat('$#,##0.00', 'D2:D10');
        spreadsheet.numberFormat('$#,##0.00', 'E2:E11');
    },
    // Triggers before going to the editing mode.
    cellEdit: (args: CellEditEventArgs): void => {
        // Preventing the editing in 5th(Amount) column.
        if (args.address.includes('E')) { args.cancel = true; }
    },
    // Trigger before saving the edited cell content.
    beforeCellSave: (args: CellEditEventArgs): void => {
        // Prevent saving the edited changes in 4th(Rate) column.
        if (args.address.includes('D')) {
            args.cancel = true;
            // Manually removes the editable state without saving the
changes. Use `endEdit` method if you want to save the changes.
            spreadsheet.closeEdit();
        }
    },
    showSheetTabs: false, showRibbon: false
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Limitations

- Text overflow in cells is not supported in Editing.

See Also

- [Cell range](#)
- [Formatting](#)
- [Hyperlink](#)
- [Undo and Redo](#)
- [Unlock the particular cells in the protected sheet](#)

Formulas in EJ2 JavaScript Spreadsheet control

Formulas are used for calculating the data in a worksheet. You can refer the cell reference from same sheet or from different sheets.

Usage

You can set formula for a cell in the following ways,

- Using the `formula` property from `cell`, you can set the formula or expression to each cell at initial load.
- Set the formula or expression through data binding.
- You can set formula for a cell by [editing](#).
- Using the [updateCell](#) method, you can set or update the cell formula.

Create User Defined Functions / Custom Functions

The Spreadsheet includes a number of built-in formulas. For your convenience, a list of supported formulas can be found [here](#).

You can define and use an unsupported formula, i.e. a user defined/custom formula, in the spreadsheet by using the [addCustomFunction](#) function. Meanwhile, remember that you should define a user defined/custom formula whose results should only return a single value. If a user-defined/custom formula returns an array, it will be time-consuming to update adjacent cell values.

The following code example shows an unsupported formula in the spreadsheet.

INDEX.TS

```
import {
  Spreadsheet,
  SheetModel,
  ColumnModel,
  RowModel,
} from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [
  { width: 150 },
```

```

    { width: 120 },
    { width: 120 },
    { width: 120 },
    { width: 140 },
    { width: 150 },
];
let rows: RowModel[] = [
    {
        height: 40,
        customHeight: true,
        cells: [
            {
                value: 'Monthly Expense',
                colSpan: 5,
                style: {
                    textAlign: 'center',
                    fontWeight: 'bold',
                    verticalAlign: 'middle',
                    fontStyle: 'italic',
                    fontSize: '15pt',
                },
            },
        ],
    },
    { height: 30 },
    {
        index: 11,
        cells: [
            { value: 'Totals', style: { fontStyle: 'italic', fontWeight: 'bold' } },
            { formula: '=SUM(B3:B11)' },
            // Calculating total of each column data through cell binding.
            { formula: '=SUM(C3:C11)' },
            { formula: '=SUM(D3:D11)' },
        ],
    },
    {
        cells: [
            {
                index: 1,
                value: 'Number of Categories',
                colSpan: 2,
                style: { fontWeight: 'bold', textAlign: 'right' },
            },
            { index: 3, formula: '=COUNTA(A3:A11)' },
        ],
    },
    {
        cells: [
            {
                index: 1,
                value: 'Average Spend',
                colSpan: 2,
                style: { fontWeight: 'bold', textAlign: 'right' },
            },
            { index: 3, formula: '=AVERAGE(B3:B11)', format: '$#,##0' },
        ],
    },
];

```

```

    },
    {
      cells: [
        {
          index: 1,
          value: 'Min Spend',
          colSpan: 2,
          style: { fontWeight: 'bold', textAlign: 'right' },
        },
        { index: 3, formula: '=MIN(B3:B11)', format: '$#,##0' },
      ],
    },
    {
      cells: [
        {
          index: 1,
          value: 'Max Spend',
          colSpan: 2,
          style: { fontWeight: 'bold', textAlign: 'right' },
        },
        { index: 3, formula: '=MAX(B3:B11)', format: '$#,##0' },
      ],
    },
  ];
  let sheets: SheetModel[] = [
    { ranges: [{ dataSource: data, startCell: 'A2' }], columns: columns, rows
  ],
  ];
  // Custom function to calculate percentage between two cell values.
  function calculatePercentage(firstCell: string, secondCell: string): number
  {
    return Number(firstCell) / Number(secondCell);
  }
  // Custom function to calculate round down for values.
  function roundDownHandler(value: number, digit: number): number {
    let multiplier: number = Math.pow(10, digit);
    return Math.floor(value * multiplier) / multiplier;
  }
  let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: (): void => {
      spreadsheet.cellFormat(
        { fontWeight: 'bold', textAlign: 'center' },
        'A2:F2'
      );
      spreadsheet.numberFormat('$#,##0', 'B3:D12');
      spreadsheet.numberFormat('0%', 'E3:E12');
      // Adding custom function for calculating the percentage between two
      // cells.
      spreadsheet.addCustomFunction(calculatePercentage, 'PERCENTAGE');
      // Adding custom function for calculating round down for the value.
      spreadsheet.addCustomFunction(roundDownHandler, 'ROUNDDOWN');
      // Calculate percentage using custom added formula in E12 cell.
      spreadsheet.updateCell({ formula: '=PERCENTAGE(C12,D12)' }, 'E12');
      // Calculate round down for average values using custom added formula in
      // F12 cell.
      spreadsheet.updateCell(

```

```

        { formula: '=ROUNDDOWN(F11,1)' },
        'F12'
    );
},
showRibbon: false,
showSheetTabs: false,
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

```



```

        <div id="container">
            <div id="spreadsheet"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Second, if you want to directly compute any formula or expression, you can use the [computeExpression](#) method. This method will work for both built-in and used-defined/custom formula.

The following code example shows how to use `computeExpression` method in the spreadsheet.

INDEX.TS

```

import {
    Spreadsheet,
    SheetModel,
    ColumnModel,
    RowModel,
} from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [
    { width: 150 },
    { width: 120 },
    { width: 120 },
    { width: 120 },
    { width: 140 },
    { width: 150 },
];
let rows: RowModel[] = [
    {
        height: 40,
        customHeight: true,
        cells: [
            {
                value: 'Monthly Expense',
                colSpan: 5,
                style: {
                    textAlign: 'center',
                    fontWeight: 'bold',
                    verticalAlign: 'middle',
                    fontStyle: 'italic',
                    fontSize: '15pt',
                },
            },
        ],
    },
    { height: 30 },
    {

```

```

        index: 11,
        cells: [
            { value: 'Totals', style: { fontStyle: 'italic', fontWeight: 'bold' } },
        ],
        { formula: '=SUM(B3:B11)' },
        // Calculating total of each column data through cell binding.
        { formula: '=SUM(C3:C11)' },
        { formula: '=SUM(D3:D11)' },
    ],
},
{
    cells: [
        {
            index: 1,
            value: 'Number of Categories',
            colSpan: 2,
            style: { fontWeight: 'bold', textAlign: 'right' },
        },
        { index: 3, formula: '=COUNTA(A3:A11)' },
    ],
},
{
    cells: [
        {
            index: 1,
            value: 'Average Spend',
            colSpan: 2,
            style: { fontWeight: 'bold', textAlign: 'right' },
        },
        { index: 3, formula: '=AVERAGE(B3:B11)', format: '$#,##0' },
    ],
},
{
    cells: [
        {
            index: 1,
            value: 'Min Spend',
            colSpan: 2,
            style: { fontWeight: 'bold', textAlign: 'right' },
        },
        { index: 3, formula: '=MIN(B3:B11)', format: '$#,##0' },
    ],
},
{
    cells: [
        {
            index: 1,
            value: 'Max Spend',
            colSpan: 2,
            style: { fontWeight: 'bold', textAlign: 'right' },
        },
        { index: 3, formula: '=MAX(B3:B11)', format: '$#,##0' },
    ],
},
    ],
};
let sheets: SheetModel[] = [

```

```

    { ranges: [{ dataSource: data, startCell: 'A2' }], columns: columns, rows
    },
  ];
  // Custom function to calculate percentage between two cell values.
  function calculatePercentage(firstCell: string, secondCell: string): number
  {
    return Number(firstCell) / Number(secondCell);
  }
  let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: (): void => {
      spreadsheet.cellFormat(
        { fontWeight: 'bold', textAlign: 'center' },
        'A2:F2'
      );
      spreadsheet.numberFormat('$#,##0', 'B3:D12');
      spreadsheet.numberFormat('0%', 'E3:E12');
      // Adding custom function for calculating the percentage between two
      cells.
      spreadsheet.addCustomFunction(calculatePercentage, 'PERCENTAGE');
      // Calculate percentage using custom added formula in E11 cell.
      spreadsheet.updateCell({ formula: '=PERCENTAGE(C11,D11)' }, 'E11');
      // Calculate expressions using computeExpression in E10 cell.
      spreadsheet.updateCell(
        { value: spreadsheet.computeExpression('C10/D10') as string },
        'E10'
      );
      // Calculate custom formula values using computeExpression in E12 cell.
      spreadsheet.updateCell(
        {
          value: spreadsheet.computeExpression('=PERCENTAGE(C12,D12)') as
string,
        },
        'E12'
      );
      // Calculate SUM (built-in) formula values using computeExpression in
      D12 cell.
      spreadsheet.updateCell(
        { value: spreadsheet.computeExpression('=SUM(D3:D11)') as string },
        'D12'
      );
    },
    showRibbon: false,
    showSheetTabs: false,
  });
  spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">

```

```

        <link rel="shortcut icon" href="resources/favicon.ico">
        <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
        <link href="styles.css" rel="stylesheet">

        <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
        <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Formula bar

Formula bar is used to edit or enter cell data in much easier way. By default, the formula bar is enabled in the spreadsheet. Use the [showFormulaBar](#) property to enable or disable the formula bar.

Named Ranges

You can define a meaningful name for a cell range and use it in the formula for calculation. It makes your formula much easier to understand and maintain. You can add named ranges to the Spreadsheet in the following ways,

- Using the [definedNames](#) collection, you can add multiple named ranges at initial load.
- Use the [addDefinedName](#) method to add a named range dynamically.
- You can remove an added named range dynamically using the [removeDefinedName](#) method.
- Select the range of cells, and then enter the name for the selected range in the **Name box**.

The following code example shows the usage of named ranges support.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel, DefineNameModel }
from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);

let columns: ColumnModel[] = [{ width: 150 }, { width: 120 }, { width: 120 }, { width: 120 }, { width: 120 }, { width: 120 }];
// Setting names for `categories`, `monthly spendings` and `annual spendings` ranges.
let definedNames: DefineNameModel[] = [
    { name: 'Categories', refersTo: '=Budget Details!A3:A11' },
    { name: 'MonthlySpendings', refersTo: '=Budget Details!B3:B11' },
    { name: 'AnnualSpendings', refersTo: '=Budget Details!C3:C11' }
];

let rows: RowModel[] = [
    {
        height: 40, customHeight: true, cells: [{ value: 'Monthly Expense', colSpan: 5, style: { textAlign: 'center', fontWeight: 'bold', verticalAlign: 'middle', fontStyle: 'italic', fontSize: '15pt' } }],
        { height: 30 },
        {
            index: 11, cells: [{ value: 'Totals', style: { fontStyle: 'italic', fontWeight: 'bold' } }, { formula: '=SUM(MonthlySpendings)' },
            // Initializing the formulas using defined names.
            { formula: '=SUM(AnnualSpendings)' }, { formula: '=SUM(LastYearSpendings)' }, { formula: '=C12/D12' } ]
        },
        {
            cells: [{ index: 1, value: 'Number of Categories', colSpan: 2, style: { fontWeight: 'bold', textAlign: 'right' } },
            { index: 3, formula: '=COUNTA(Categories)' } ]
        },
        {
            cells: [{ index: 1, value: 'Average Spend', colSpan: 2, style: { fontWeight: 'bold', textAlign: 'right' } },
            { index: 3, formula: '=AVERAGE(MonthlySpendings)', format: '$#,##0' } ]
        },
        {

```

```

        cells: [{ index: 1, value: 'Min Spend', colSpan: 2, style: {
fontWeight: 'bold', textAlign: 'right' } }],
        { index: 3, formula: '=MIN(MonthlySpending)', format: '$#,##0'
    }]
    },
    {
        cells: [{ index: 1, value: 'Max Spend', colSpan: 2, style: {
fontWeight: 'bold', textAlign: 'right' } }],
        { index: 3, formula: '=MAX(MonthlySpending)', format: '$#,##0'
    }]
    }
];
let sheets: SheetModel[] = [{ name: 'Budget Details', ranges: [{ dataSource:
data, startCell: 'A2' }], columns: columns, rows }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    definedNames: definedNames,
    beforeDataBound: (): void => {
        // Adding name dynamically for `last year spending` and `percentage
change` ranges.
        spreadsheet.addDefinedName({ name: 'LastYearSpending', refersTo:
'=D3:D11' });
        spreadsheet.addDefinedName({ name: 'PercentageChange', refersTo:
'=E3:E11' });
    },
    created: (): void => {
        // Removing the unwanted `PercentageChange` named range
        spreadsheet.removeDefinedName('PercentageChange', '');
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A2:E2');
        spreadsheet.numberFormat('$#,##0', 'B3:D12');
        spreadsheet.numberFormat('0%', 'E3:E12');
    },
    showRibbon: false, showSheetTabs: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Supported Formulas

The following are the list of formulas supported in spreadsheet,

Formula	Description
---------	-------------

-----	-----
-------	-------

ABS	Returns the value of a number without its sign.
-----	---

ADDRESS	Returns a cell reference as text, given specified row and column numbers.
---------	---

AND	Returns TRUE if all the arguments are TRUE, otherwise returns FALSE.
-----	--

AVERAGE	Calculates average for the series of numbers and/or cells excluding text.
---------	---

AVERAGEA	Calculates the average for the cells evaluating TRUE as 1, text and FALSE as 0.
----------	---

AVERAGEIF	Clears content of the active cell and enables edit mode.
-----------	--

| AVERAGEIFS | Calculates average for the cells based on specified conditions. |

| CEILING | Rounds a number up to the nearest multiple of a given factor. |

| CHOOSE | Returns a value from list of values, based on index number. |

| CHAR | Returns the character from the specified number. |

| CODE | Returns the numeric code for the first character in a given string. |

| CONCAT | Concatenates a list or a range of text strings. |

| CONCATENATE | Combines two or more strings together. |

| COUNT | Counts the cells that contain numeric values in a range. |

| COUNTA | Counts the cells that contains values in a range. |

| COUNTBLANK | Returns the number of empty cells in a specified range of cells. |

| COUNTIF | Counts the cells based on specified condition. |

| COUNTIFS | Counts the cells based on specified conditions. |

| DATE | Returns the date based on given year, month, and day. |

| DATEVALUE | Converts a date string into date value. |

| DAY | Returns the day from the given date. |

| DAYS | Returns the number of days between two dates. |

| DECIMAL | Converts a text representation of a number in a given base into a decimal number. |

| DEGREES | Converts radians to degrees. |

| DOLLAR | Converts the number to currency formatted text. |

| EDATE | Returns a date with given number of months before or after the specified date. |

| EOMONTH | Returns the last day of the month that is a specified number of months before or after an initially supplied start date. |

| EVEN | Rounds a positive number up and negative number down to the nearest even integer. |

| EXACT | Checks whether a two text strings are exactly same and returns TRUE or FALSE. |

| EXP | Returns e raised to the power of the given number. |

| FACT | Returns the factorial of a number. |

| FIND | Returns the position of a string within another string, which is case sensitive. |

| FLOOR | Rounds a number down to the nearest multiple of a given factor. |

| HLOOKUP | Looks for a value in the top row of the array of values and then returns a value in the same column from a row in the array that you specify. |

| HOUR | Returns the number of hours in a specified time string. |

| IF | Returns value based on the given expression. |

| IFERROR | Returns value if no error found else it will return specified value. |

| IFS | Returns value based on the given multiple expressions. |

| INDEX | Returns a value of the cell in a given range based on row and column number. |

| INT | Rounds a number down to the nearest integer. |

| INTERCEPT | Calculates the point of the Y-intercept line via linear regression. |

| ISNUMBER | Returns true when the value parses as a numeric value. |

| LARGE | Returns the **k-th** largest value in a given array. |

| LEN | Returns a number of characters in a given string. |

| LN | Returns the natural logarithm of a number. |

| LOG | Returns the logarithm of a number to the base that you specify. |

| LOOKUP | Looks for a value in a one-row or one-column range, then returns a value from the same position in a second one-row or one-column range. |

| MATCH | Returns the relative position of a specified value in given range. |

| MAX | Returns the largest number of the given arguments. |

| MEDIAN | Returns the median of the given set of numbers. |

| MINUTE | Returns the number of minutes in a specified time string. |

| MIN | Returns the smallest number of the given arguments. |

| MOD | Returns a remainder after a number is divided by divisor. |

| MONTH | Returns the number of months in a specified date string. |

| NOT | Returns the inverse of a given logical expression. |

| NOW | Returns the current date and time. |

| ODD | Rounds a positive number up and negative number down to the nearest odd integer. |

| OR | Returns TRUE if any of the arguments are TRUE, otherwise returns FALSE. |

| PI | Returns the value of pi. |

| POWER | Returns the result of a number raised to power. |

| PRODUCT | Multiplies a series of numbers and/or cells. |

| RADIANS | Converts degrees into radians. |

| RAND | Returns a random number between 0 and 1. |

| RANDBETWEEN | Returns a random integer based on specified values. |

| ROUND | Rounds a number to the specified number of digits. |

| ROUNDDOWN | Rounds a number down, toward zero. |

| ROUNDUP | Rounds a number up, away from zero. |

| RSQ | Returns the square of the Pearson product moment correlation coefficient based on data points in *knowny's and knownx's*. |

| SECOND | Returns the number of seconds in a specified time string. |

| SMALL | Returns the **k-th** smallest value in a given array. |

- | SLOPE | Returns the slope of the line from linear regression of the data points. |
- | SORT | Sorts the contents of a column, range, or array in ascending or descending order. |
- | SQRT | Returns the square root of a positive number. |
- | SUBTOTAL | Returns subtotal for a range using the given function number. |
- | SUM | Adds a series of numbers and/or cells. |
- | SUMIF | Adds the cells based on specified condition. |
- | SUMIFS | Adds the cells based on specified conditions. |
- | SUMPRODUCT | Returns the sum of the products of the corresponding array in given arrays. |
- | T | Checks whether a value is text or not and returns the text. |
- | TEXT | Converts the supplied value into text by using the user-specified format. |
- | TIME | Converts hours, minutes, seconds to the time formatted text. |
- | TODAY | Returns the current date. |
- | TRUNC | Truncates a supplied number to a specified number of decimal places. |
- | UNIQUE | Returns a unique values from a range or array. |
- | VLOOKUP | Looks for a specific value in the first column of a lookup range and returns a corresponding value from a different column within the same row. |

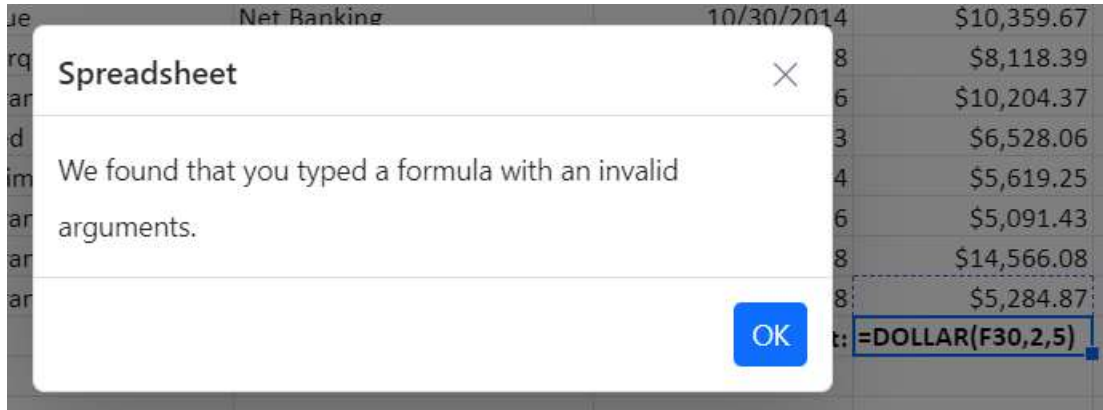
Formula Error Dialog

If you enter an invalid formula in a cell, an error dialog with an error message will appear. For instance, a formula with the incorrect number of arguments, a formula without parenthesis, etc.

Error Message	Reason
----- -----	
We found that you typed a formula with an invalid arguments	Occurs when passing an argument even though it wasn't needed.
We found that you typed a formula with an empty expression	Occurs when passing an empty expression in the argument.
We found that you typed a formula with one or more missing opening or closing parenthesis	Occurs when an open parenthesis or a close parenthesis is missing.
We found that you typed a formula which is improper	Occurs when passing a single reference but a range was needed.
We found that you typed a formula with a wrong number of arguments	Occurs when the required arguments were not passed.
We found that you typed a formula which requires 3 arguments	Occurs when the required 3 arguments were not passed.
We found that you typed a formula with a mismatched quotes	Occurs when passing an argument with mismatched quotes.

| We found that you typed a formula with a circular reference | Occurs when passing a formula with circular cell reference. |

| We found that you typed a formula which is invalid | Except in the cases mentioned above, all other errors will fall into this broad category. |



See Also

- [Editing](#)
- [Formatting](#)
- [Open](#)
- [Save](#)

Formatting in EJ2 JavaScript Spreadsheet control

Formatting options make your data easier to view and understand. The different types of formatting options in the Spreadsheet are,

- Number Formatting
- Text Formatting
- Cell Formatting

Number Formatting

Number formatting provides a type for your data in the Spreadsheet. Use the [allowNumberFormatting](#) property to enable or disable the number formatting option in the Spreadsheet. The different types of number formatting supported in Spreadsheet are,

| Types | Format |

|-----|-----|

| General(default) | NA |

| Number | 0.00 |

| Currency | \$#,##0.00 |

| Accounting | (\$ #,##0.00);(\$ (#,##0.00);(\$* "-"??);(@_) |

| ShortDate | mm-dd-yyyy |

LongDate	dddd, mmmm dd, yyyy
Time	h:mm:ss AM/PM
Percentage	0.00%
Fraction	# ?/?
Scientific	0.00E+00
Text	@

Number formatting can be applied in following ways,

- Using the `format` property in `cell`, you can set the desired format to each cell at initial load.
- Using the `numberFormat` method, you can set the number format to a cell or range of cells.
- Selecting the number format option from ribbon toolbar.

Custom Number Formatting

Spreadsheet supports custom number formats to display your data as numbers, dates, times, percentages, and currency values. If the pre-defined number formats do not meet your needs, you can set your own custom formats using custom number formats dialog or `numberFormat` method.

The different types of custom number formatting supported in Spreadsheet are,

Types	Format
General(default)	NA
Number	0
Number	0.00
Number	#,##0
Number	#,##0.00
Number	#,##0_);(##,##0)
Number	`##,##0_);Red`
Number	#,##0.00_);(##,##0.00)
Number	`##,##0.00_);Red`
Currency	\$##,##0_);(\$##,##0)
Currency	`\$##,##0_);Red`
Currency	\$##,##0.00_);(\$##,##0.00)
Currency	`\$##,##0.00_);Red`
Percentage	0%
Percentage	0.00%

Scientific	0.00E+00
Scientific	##0.0E+0
Fraction	# ?/?
Fraction	# ??/??
ShortDate	dd-mm-yy
Custom	dd-mmm-yy
Custom	dd-mmm
Custom	mmm-yy
Custom	h:mm AM/PM
Custom	h:mm:ss AM/PM
Custom	h:mm
Custom	h:mm:ss
Custom	dd-mm-yy h:mm
Custom	mm:ss
Custom	mm:ss.0
Text	@
Custom	[h]:mm:ss
Accounting	(\$ #,##0);(\$ (#,##0);(\$* "-");(@_)
Accounting	(#,##0);((#,##0);(* "-");(@_)
Accounting	(\$ #,##0.00);(\$ (#,##0.00);(\$* "-"?);(@_)
Accounting	(#,##0.00);((#,##0.00);(* "-"?);(@_)

Custom Number formatting can be applied in following ways,

- Using the [numberFormat](#) method, you can set your own custom number format to a cell or range of cells.
- Selecting the custom number format option from custom number formats dialog or type your own format in dialog input and then click apply button. It will apply the custom format for selected cells.

The following code example shows the number formatting in cell data.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel, getFormatFromType,
NumberFormatType } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
```

```

let columns: ColumnModel[] = [{ width: 140 }, { width: 140 }, { width: 160 }, { width: 160 }, { width: 160 }, { width: 120 }];
let rows: RowModel[] = [{
    height: 35, customHeight: true,
    cells: [{ value: 'Sales Team Summary', colSpan: 6,
        style: { verticalAlign: 'middle', textAlign: 'center', fontSize: '16pt',
fontWeight: 'bold',
            border: '1px solid #e0e0e0', backgroundColor: '#EEEEEE', color: '#279377' } }],
},
{
    index: 10, cells: [{ index: 1, value: 'Total:', style: { fontWeight: 'bold', fontStyle: 'italic' } },
        // If the format string is not known, you can get the format string
        // using `getFormatFromType` function like below
        { formula: '=SUM(C3:C10)', format:
getFormatFromType(<NumberFormatType>'Accounting') },
        // Applied number format to C11, D11 & E11 through cell binding
        { formula: '=SUM(D3:D10)', format: '_($* #,##0.00_);_($* (#,##0.00);_($*
"-"??_);_(@_) ' },
        { formula: '=SUM(E3:E10)', format: '_($* #,##0.00_);_($* (#,##0.00);_($*
"-"??_);_(@_) ' } ]
    }];
let sheets: SheetModel[] = [{
    ranges: [{ dataSource: data, startCell: 'A2' }],
    columns: columns, rows: rows, showGridLines: false, selectedRange: 'U15'
}];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    // Applied number formatting to the range of cells using 'numberFormat'
    // method once the component is loaded
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', fontSize: '12pt',
            backgroundColor: '#279377', textAlign: 'center', color: '#ffffff',
            borderBottom: '1px solid #e0e0e0' }, 'A2:F2');
        spreadsheet.cellFormat({ borderTop: '1px solid #e0e0e0',
            backgroundColor: '#EEEEEE' }, 'A11:F11');
        spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A2:F11',
            'Outer');
        // Applied Accounting format to the cells from C3 to E10 range.
        spreadsheet.numberFormat('_($* #,##0.00_);_($* (#,##0.00);_($* "-"
"??_);_(@_) ', 'C3:E10');
        // Applied Percentage format to the cells from C3 to E11 range.
        spreadsheet.numberFormat('0%', 'F3:F10');
        // applied the custom number format for cell form D3 to D10 range
        spreadsheet.numberFormat('[Red] [<=2000] $#,##0.00; [Blue] [>2000] $#,##0.00',
            'D3:D10');
        // applied the custom number format for cell from F3 to F10 range
        spreadsheet.numberFormat('#,##0.00_); [Red] (#,##0.00) ', 'F3:F10');
    },
    // Removed the unwanted support for this samples
    showRibbon: false, showFormulaBar: false, showSheetTabs: false,
    allowInsert: false, allowDelete: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Spreadsheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Text and cell formatting

Text and cell formatting enhances the look and feel of your cell. It helps to highlight a particular cell or range of cells from a whole workbook. You can apply formats like font size, font family, font color, text alignment, border etc. to a cell or range of cells. Use the [allowCellFormatting](#) property to enable or disable the text and cell formatting option in Spreadsheet. You can set the formats in following ways,

- Using the `style` property, you can set formats to each cell at initial load.
- Using the [cellFormat](#) method, you can set formats to a cell or range of cells.
- You can also apply by clicking the desired format option from the ribbon toolbar.

Fonts

Various font formats supported in the spreadsheet are font-family, font-size, bold, italic, strike-through, underline and font color.

Text Alignment

You can align text in a cell either vertically or horizontally using the `textAlign` and `verticalAlign` property.

Indents

To enhance the appearance of text in a cell, you can change the indentation of a cell content using `textIndent` property.

Fill color

To highlight cell or range of cells from whole workbook you can apply background color for a cell using `backgroundColor` property.

Borders

You can add borders around a cell or range of cells to define a section of worksheet or a table. The different types of border options available in the spreadsheet are,

| Types | Actions |

|-----|-----|

| Top Border | Specifies the top border of a cell or range of cells.|

| Left Border | Specifies the left border of a cell or range of cells.|

| Right Border | Specifies the right border of a cell or range of cells.|

| Bottom Border | Specifies the bottom border of a cell or range of cells.|

| No Border | Used to clear the border from a cell or range of cells.|

| All Border | Specifies all border of a cell or range of cells.|

| Horizontal Border | Specifies the top and bottom border of a cell or range of cells.|

| Vertical Border | Specifies the left and right border of a cell or range of cells.|

| Outside Border | Specifies the outside border of a range of cells.|

| Inside Border | Specifies the inside border of a range of cells.|

You can also change the color, size, and style of the border. The size and style supported in the spreadsheet are,

| Types | Actions |

|-----|-----|

| Thin | Specifies the **1px** border size (default).|

| Medium | Specifies the **2px** border size.|

| Thick | Specifies the **3px** border size.|

| Solid | Used to create the **solid** border (default).|

| Dashed | Used to create the **dashed** border.|

| Dotted | Used to create the **dotted** border.|

| Double | Used to create the **double** border.|

Borders can be applied in the following ways,

- Using the **border**, **borderLeft**, **borderRight**, **borderBottom** properties, you can set the desired border to each cell at initial load.
- Using the **setBorder** method, you can set various border options to a cell or range of cells.
- Selecting the border options from ribbon toolbar.

The following code example shows the style formatting in text and cells of the spreadsheet.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel, CellStyleModel }
from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 100 }, { width: 200 }, { width: 110 }, { width: 140 }, { width: 90 }];
let rows: RowModel[] = [{
    height: 40, customHeight: true,
    cells: [{ value: 'Order Summary', colSpan: 5,
        // Applied styles to 'A1' cell through cell property binding
        style: { fontFamily: 'Axettac Demo', verticalAlign: 'middle',
        textAlign: 'center', fontSize: '18pt', fontWeight: 'bold', color: '#279377',
        border: '1px solid #e0e0e0' }
    } ] },
    {
        // Applied styles to 'C2' cell through cell property binding
        height: 30, cells: [{ index: 2, style: { textAlign: 'right' } } ]
    },
    { height: 30 }, { height: 30 }, { height: 30 }, { height: 30 }, {
height: 30 },
    { height: 30 }, { height: 30 }, { height: 30 }, { height: 30 }, {
height: 30 }];
let sheets: SheetModel[] = [{
    ranges: [{ dataSource: data, startCell: 'A2' }],
```

```

        columns: columns, rows: rows, showGridLines: false, selectedRange: 'U15'
    }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    // Applied cell formatting to the range of cells using 'cellFormat'
    // method once the component is loaded
    created: (): void => {
        // Setting common styles to table header cells
        spreadsheet.cellFormat({ fontWeight: 'bold', fontSize: '12pt',
background-color: '#279377', color: '#ffffff' }, 'A2:E2');
        // Setting common styles to whole table cells
        spreadsheet.cellFormat({ verticalAlign: 'middle', fontFamily:
'Axettac Demo' }, 'A2:E12');
        // Column wise styles setting
        spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:A12');
        // Setting text-indent to 2 and 4 column
        let style: CellStyleModel = { textAlign: 'left', textIndent: '8pt'
    };

        spreadsheet.cellFormat(style, 'B2:B12');
        spreadsheet.cellFormat(style, 'D2:D12');
        spreadsheet.cellFormat({ fontStyle: 'italic', textAlign: 'right' },
'C3:C12');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'E2:E12');
        // Applied border to range of cells using 'setBorder' method
        spreadsheet.setBorder({ borderLeft: '1px solid #e0e0e0',
borderRight: '1px solid #e0e0e0' }, 'A2:E2');
        spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A4:E11',
'Horizontal');
        spreadsheet.setBorder({ border: '1px solid #e0e0e0' }, 'A3:E12',
'Outer');
        spreadsheet.cellFormat({ color: '#10c469', textDecoration: 'line-
through' }, 'E3:E4');
        spreadsheet.cellFormat({ color: '#10c469', textDecoration: 'line-
through' }, 'E9');
        spreadsheet.cellFormat({ color: '#10c469', textDecoration: 'line-
through' }, 'E12');
        spreadsheet.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E5');
        spreadsheet.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E8');
        spreadsheet.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E11');
        spreadsheet.cellFormat({ color: '#62c9e8' }, 'E6');
        spreadsheet.cellFormat({ color: '#62c9e8' }, 'E10');
        spreadsheet.cellFormat({ color: '#ff5b5b' }, 'E7');
    },
    // Removed the unwanted support for this samples
    showRibbon: false, showFormulaBar: false, showSheetTabs: false,
allowEditing: false, allowInsert: false, allowDelete: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>

```

```

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Formatting

The following features are not supported in Formatting:

- Insert row/column between the formatting applied cells.
- Formatting support for row/column.

Conditional Formatting

Conditional formatting helps you to format a cell or range of cells based on the conditions applied. You can enable or disable conditional formats by using the [allowConditionalFormat](#) property.

The default value for the `allowConditionalFormat` property is `true`.

Apply Conditional Formatting

You can apply conditional formatting by using one of the following ways,

- Select the conditional formatting icon in the Ribbon toolbar under the Home Tab.
- Using the [conditionalFormat\(\)](#) method to define the condition.
- Using the `conditionalFormats` in sheets model.

Conditional formatting has the following types in the spreadsheet,

Highlight cells rules

Highlight cells rules option in the conditional formatting enables you to highlight cells with a preset color depending on the cell's value.

The following options can be given for the highlight cells rules as type,

'GreaterThan', 'LessThan', 'Between', 'EqualTo', 'ContainsText', 'DateOccur', 'Duplicate', 'Unique'.

The following preset colors can be used for formatting styles,

`"RedFT"` - Light Red Fill with Dark Red Text,

`"YellowFT"` - Yellow Fill with Dark Yellow Text,

`"GreenFT"` - Green Fill with Dark Green Text,

`"RedF"` - Red Fill,

`"RedT"` - Red Text.

Top bottom rules

Top bottom rules option in the conditional formatting allows you to apply formatting to the cells that satisfy a statistical condition with other cells in the range.

The following options can be given for the top bottom rules as type,

'Top10Items', 'Bottom10Items', 'Top10Percentage', 'Bottom10Percentage', 'BelowAverage', 'AboveAverage'.

Data Bars

You can apply data bars to represent the data graphically inside a cell. The longest bar represents the highest value and the shorter bars represent the smaller values.

The following options can be given for the data bars as type,

'BlueDataBar', 'GreenDataBar', 'RedDataBar', 'OrangeDataBar', 'LightBlueDataBar', 'PurpleDataBar'.

Color Scales

Using color scales, you can format your cells with two or three colors, where different color shades represent the different cell values. In the Green-Yellow-Red(GYR) Color Scale, the cell that holds the minimum value is colored as red. The cell that holds the median is colored as yellow, and the cell that holds the maximum value is colored as green. All other cells are colored proportionally.

The following options can be given for the color scales as type,

'GYRColorScale', 'RYGColorScale', 'GWRCColorScale', 'RWGColorScale', 'BWRColorScale', 'RWBColorScale', 'WRCColorScale', 'RWColorScale', 'GWColorScale', 'WGColorScale', 'GYColorScale', 'YGColorScale'.

Icon Sets

Icon sets will help you to visually represent your data with icons. Every icon represents a range of values. In the Three Arrows(colored) icon, the green arrow icon represents the values greater than 67%, the yellow arrow icon represents the values between 33% to 67%, and the red arrow icon represents the values less than 33%.

The following options can be given for the icon sets as type,

'ThreeArrows', 'ThreeArrowsGray', 'FourArrowsGray', 'FourArrows', 'FiveArrowsGray', 'FiveArrows', 'ThreeTrafficLights1', 'ThreeTrafficLights2', 'ThreeSigns', 'FourTrafficLights', 'FourRedToBlack', 'ThreeSymbols', 'ThreeSymbols2', 'ThreeFlags', 'FourRating', 'FiveQuarters', 'FiveRating', 'ThreeTriangles', 'ThreeStars', 'FiveBoxes'.

Custom Format

Using the custom format for conditional formatting you can set cell styles like color, background color, font style, font weight, and underline.

In the MAY and JUN columns, we have applied conditional formatting custom format.

In the Conditional format, custom format supported for Highlight cell rules and Top bottom rules.

Clear Rules

You can clear the defined rules by using one of the following ways,

- Using the “Clear Rules” option in the Conditional Formatting button of HOME Tab in the ribbon to clear the rule from selected cells.
- Using the [clearConditionalFormat\(\)](#) method to clear the defined rules.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
 '@syncfusion/ej2-spreadsheet';
import { conditionalFormatData } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [{
  name: 'Car Sales Record',
  ranges: [{ dataSource: conditionalFormatData }],
  conditionalFormats: [
    { type: 'GreaterThan', cFColor: 'RedFT', value: '700', range:
    'B2:B9' },
```

```

        { type: 'Bottom10Items', cFColor: 'YellowFT', value: '4', range:
'C2:C9' },
        { type: 'BlueDataBar', range: 'D2:D9' }
    ],
    columns: [{ width: 120 }]]];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    // Removed the unwanted support for this samples
    showFormulaBar: false,
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:N1');
        spreadsheet.conditionalFormat({ type: "RYGColorScale", range:
'E2:E9' });
        spreadsheet.conditionalFormat({ type: "ThreeArrows", range: 'H2:H9'
});
        //Custom format
        spreadsheet.conditionalFormat({ type: 'Top10Items', value: '1',
            format: { style: { color: '#ffffff', backgroundColor: '#009999',
fontWeight: 'bold' }}, range: 'F2:F9' });
        spreadsheet.conditionalFormat({ type: 'Bottom10Items', value: '1',
            format: { style: { color: '#ffffff', backgroundColor: '#c68d53',
fontWeight: 'bold' }}, range: 'G2:G9' });
    }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Conditional formatting

The following features have some limitations in Conditional Formatting:

- Insert row/column between the conditional formatting.
- Conditional formatting with formula support.
- Copy and paste the conditional formatting applied cells.
- Custom rule support.

See Also

- [Rows and columns](#)
- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)
- [Ribbon customization](#)

Freeze pane in EJ2 JavaScript Spreadsheet control

Freeze Panes helps you to keep particular rows or columns visible when scrolling the sheet content in the spreadsheet. You can specify the number of frozen rows and columns using the [frozenRows](#) and [frozenColumns](#) properties inside the [Sheet](#) property.

Apply freezepanes on UI

User Interface:

In the active spreadsheet, click the cell where you want to create freeze panes. Freeze panes can be done in any of the following ways:

- Select the View tab in the Ribbon toolbar and choose the **Freeze Panes** item.
- Use the [freezePanes](#) method programmatically.

FrozenRows

It allows you to keep a certain number of rows visible while scrolling vertically through the rest of the worksheet.

User Interface:

In the active spreadsheet, select the cell where you want to create frozen rows. Frozen rows can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the **Freeze Rows** item.
- You can specify the number of frozen rows using the **frozenRows** property inside the **Sheet** property.

FrozenColumns

It allows you to keep a certain number of columns visible while scrolling horizontally through the rest of the worksheet.

User Interface:

In the active spreadsheet, select the cell where you want to create frozen columns. Frozen columns can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the **Freeze Columns** item.
- You can specify the number of frozen columns using the **frozenColumns** property inside the **Sheet** property.

In this demo, the frozenColumns is set as '2', and the frozenRows is set as '2'. Hence, the two columns on the left and the top two rows are frozen.

INDEX.TS

```
import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import { tradeData } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 180 }, { width: 180 }, { width: 180 },
    { width: 180 }, { width: 180 }, { width: 180 },
    { width: 180 }, { width: 180 }, { width: 180 }, { width: 180 }, { width:
180 }, { width: 180}]
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ ranges: [{ dataSource: tradeData }], columns: columns
        // Specifies the number of frozen rows
        frozenRows: 2,
        // Specifies the number of frozen columns
        frozenColumns: 2
    }],
});
```



```
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations

Here, we have listed out the limitations with Freeze Panes feature.

- Merging the cells between freeze and unfreeze area.
- If images and charts are added inside the freeze area cells, their portion in the unfreeze area will not move when scrolling.

See Also

- [Sorting](#)
- [Filtering](#)
- [Undo Redo](#)

Context menu in EJ2 JavaScript Spreadsheet control

Context Menu is used to improve user interaction with Spreadsheet using the popup menu. This will open when right-clicking on Cell/Column Header/Row Header/ Pager in the Spreadsheet. You can use [enableContextMenu](#) property to enable/disable context menu.

The default value for the `enableContextMenu` property is `true`.

Context Menu Items in Row Cell

Please find the table below for default context menu items and their actions.

Context Menu items	Action
Cut	Cut the selected cells data to the clipboard, you can select a cell where you want to move the data.
Copy	Copy the selected cells data to the clipboard, so that you can paste it to somewhere else.
Paste	Paste the data from clipboard to spreadsheet.
Paste Special	Values - Paste the data values from clipboard to spreadsheet. Formats - Paste the data formats from clipboard to spreadsheet.
Filter	Perform filtering to the selected cells based on an active cell's value.
Sort	Perform sorting to the selected range of cells by ascending or descending.
Hyperlink	Create a link in the spreadsheet to navigate to web links or cell reference within the sheet or other sheets in the Spreadsheet.

Context Menu Items in Row Header / Column Header

Please find the table below for default context menu items and their actions.

Context Menu items	Action
-----	-----

| [Cut](#) | Cut the selected row/column header data to the clipboard, you can select a cell where you want to move the data. |

| [Copy](#) | Copy the selected row/column header data to the clipboard, so that you can paste it to somewhere else. |

| [Paste](#) | Paste the data from clipboard to spreadsheet. |

| [Paste Special](#) | **Values** - Paste the data values from clipboard to spreadsheet. **Formats** - Paste the data formats from clipboard to spreadsheet. |

| [Insert Columns](#) | Insert new rows or columns into the worksheet. |

| [Delete Columns](#) | Delete existing rows or columns from the worksheet. |

| [Hide Columns](#) | Hide the rows and columns. |

| [UnHide Columns](#) | Show the hidden rows and columns. |

Context Menu Items in Pager

Please find the table below for default context menu items and their actions.

Context Menu items	Action
----- -----	
Insert	Insert a new worksheet in front of an existing worksheet in the spreadsheet.
Delete	Delete the selected worksheet from the spreadsheet.
Rename	Rename the selected worksheet.
Protect Sheet	Prevent unwanted changes from others by limiting their ability to edit.
Hide	Hide the selected worksheet.

Context Menu Customization

You can perform the following context menu customization options in the spreadsheet

- Add Context Menu Items
- Remove Context Menu Items
- Enable/Disable Context Menu Items

Add Context Menu Items

You can add the custom items in context menu using the [addContextMenuItems](#) in **contextmenuBeforeOpen** event

In this demo, Custom Item is added after the Paste item in the context menu.

INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
//Initialize Spreadsheet component
let spreadsheet: Spreadsheet = new Spreadsheet({
  contextMenuBeforeOpen: (args): void => {
    if (args.element.id === spreadsheet.element.id + '_contextmenu')
    {
      // To add context menu items.
    }
  }
});
```

```

        spreadsheet.addContextMenuItems([ { text: 'Custom Item' } ],
        'Paste Special', false);
        //To pass the items, Item before / after that the element to be
        inserted, Set false if the items need to be inserted before the text.
    }
}
});
//Render initialized Spreadsheet component
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

```

```

        <div id="container">
            <div id="spreadsheet"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Remove Context Menu Items

You can remove the items in context menu using the [removeContextMenuItems](#) in `contextmenuBeforeOpen` event

In this demo, Insert Column item has been removed from the row/column header context menu.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
//Initialize Spreadsheet component
let spreadsheet: Spreadsheet = new Spreadsheet({
    // To remove existing context menu items.
    contextMenuBeforeOpen: (): void => {
        spreadsheet.removeContextMenuItems(["Insert Column"], false);
    }
});
//Items that needs to be removed, Set `true` if the given `text` is a unique id.
//Render initialized Spreadsheet component
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Enable/Disable Context Menu Items

You can enable/disable the items in context menu using the [enableContextMenuItems](#) in `contextmenuBeforeOpen` event

In this demo, Rename item is disabled in the pager context menu.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
//Initialize Spreadsheet component
let spreadsheet: Spreadsheet = new Spreadsheet({
  contextMenuBeforeOpen: (): void => {
    //To enable / disable context menu items.
    spreadsheet.enableContextMenuItems(['Rename'], false, false);
    // Contextmenu Items that needs to be enabled / disabled, Set true
    / false to enable / disable the menu items, Set true if the given text is a
    unique id.
  }
});
//Render initialized Spreadsheet component
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Worksheet](#)
- [Rows and columns](#)

Template in EJ2 JavaScript Spreadsheet control

Cell Template is used for adding HTML elements into Spreadsheet. You can add the cell template in spreadsheet by using the `template` property and specify the address using the `address` property inside the `ranges` property. You can customize the HTML elements similar to Syncfusion components (TextBox, DropDownList, RadioButton, MultiSelect, DatePicker etc) by using the `beforeCellRender` event. In this demo, Cell template is applied to C2:C9 and instantiated with HTML input components like TextBox, RadioButton, TextArea. You need to bind the events to perform any operations through HTML elements or Syncfusion components. Here, we have added `change` event in to the MultiSelect control, and we have updated the selected data into the spreadsheet cell through that change event.

The following code example describes the above behavior.

INDEX.TS

```
import { TextBox } from '@syncfusion/ej2-inputs';
import { RadioButton } from '@syncfusion/ej2-buttons';
import { DatePicker } from '@syncfusion/ej2-calendars';
import { DropDownList, MultiSelect } from '@syncfusion/ej2-dropdowns';
import { Spreadsheet, CellRenderEventArgs, BeforeSelectEventArgs,
getRangeIndexes, ChangeEventArgs } from '@syncfusion/ej2-spreadsheet';
/**
 * Cell template
 */
//Initialize Spreadsheet component
let spreadsheet: Spreadsheet = new Spreadsheet({
  showRibbon: false,
  showFormulaBar: false,
  allowOpen: false,
  allowSave: false,
  allowEditing: false,
  selectionSettings: { mode: 'None' },
  scrollSettings: {
    isFinite: true
  },
  sheets: [{
    rowCount: 40,
    showGridLines: false,
    name: 'Registration Form',
    rows: [{
      height: 55,
      cells: [{
        index: 1,
        value: 'Interview Registration Form',
        style: {
          fontSize: '12pt',
          fontWeight: 'bold',
          textAlign: 'center',
          verticalAlign: 'middle',
          textDecoration: 'underline'
        }
      }]
    }]
  }]
});
```



```

    }
  }, {
    height: 45,
    cells: [{
      index: 1,
      value: 'Name:'
    }],
  }, {
    height: 45,
    cells: [{
      index: 1,
      value: 'Date of Birth:'
    }],
  }, {
    height: 45,
    cells: [{
      index: 1,
      value: 'Gender:'
    }],
  }, {
    height: 45,
    cells: [{
      index: 1,
      value: 'Year of Experience:'
    }],
  }, {
    height: 45,
    cells: [{
      index: 1,
      value: 'Areas of Interest:'
    }],
  }, {
    height: 45,
    cells: [{
      index: 1,
      value: 'Mobile Number:'
    }],
  }, {
    height: 45,
    cells: [{
      index: 1,
      value: 'Email:'
    }],
  }, {
    height: 82,
    cells: [{
      index: 1,
      value: 'Address:'
    }],
  },
],
columns: [{
  index: 1,
  width: 190
}, {
  width: 350
}],

```

```

        ranges: [{
            template: '<input />',
            address: 'C2:C3'
        }, {
            template: '<div><input type="radio" name="gender"
value="male" /><input type="radio" name="gender" value="female"/></div>',
            address: 'C4'
        }, {
            template: '<input />',
            address: 'C5:C8'
        }, {
            template: '<textarea rows="3"/>',
            address: 'C9'
        }, {
            template: '<button class="e-btn e-flat"
style="float:right">Add</button>',
            address: 'C11'
        }
    ]
},
beforeCellRender: (args: CellRenderEventArgs) => {
    //Initializing input components before cell rendering
    if (spreadsheet.activeSheetIndex === 0) {
        let target: HTMLInputElement =
args.element.firstElementChild as HTMLInputElement;
        switch (args.address) {
            case 'B1':
                (args.element as HTMLTableCellElement).colSpan = 2;
                break;
            case 'C2':
                new TextBox({ placeholder: 'Name' }, target);
                break;
            case 'C3':
                new DatePicker({ placeholder: 'DOB', }, target);
                break;
            case 'C4':
                new RadioButton({ label: 'Male' },
args.element.firstElementChild.firstElementChild as HTMLInputElement);
                new RadioButton({ label: 'Female' },
args.element.firstElementChild.lastElementChild as HTMLInputElement);
                break;
            case 'C5':
                let experience: string[] = ['0 - 1 year', '1 - 3
years', '3 - 5 years', '5 - 10 years'];
                new DropDownList(
                    { placeholder: 'Experience', dataSource:
experience}, target );
                break;
            case 'C6':
                let languages: string[] = ['JAVA', 'C#', 'SQL'];
                new MultiSelect(
                    { showClearButton: false, placeholder: 'Areas of
Interest', dataSource: languages, change: (evt: ChangeEventArgs) => {
                        if (args.cell) {
                            args.cell.value = evt.value;
                        } else {
                            let range: number[] =
getRangeIndexes(evt.address);

```

```

spreadsheet.sheets[spreadsheet.activeSheetIndex].rows[range[0]].cells[range[
1]] = { value: evt.value };
    }
    } }, target);
    break;
    case 'C7':
        new TextBox({ placeholder: 'Mobile Number' },
target);

        break;
    case 'C8':
        new TextBox({ placeholder: 'Email'}, target);
        break;
    case 'C9':
        new TextBox(null, target);
        break;
    }
    }
    },
    created: () => {
        //Applies format to specified range
        spreadsheet.cellFormat({ fontWeight: 'bold' }, 'B2:B9');
    },
    beforeSelect: (args: BeforeSelectEventArgs) => {
        //Prevents selection
        args.cancel = true;
    }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Worksheet](#)
- [Rows and columns](#)

Illustrations in EJ2 JavaScript Spreadsheet control

Illustrations help you to insert an image, shapes, and graphic objects in the Essential JS 2 spreadsheet.

Image

Adding images to a spreadsheet can enhance the visual appeal and help to convey information more clearly.

The default value for the `allowImage` property is `true`.

Insert Image

You can insert the image by using one of the following ways,

- Selecting the Insert tab in the Ribbon toolbar, and then choose the Image tab.
- Use the `insertImage()` method programmatically.

The available parameters in `insertImage()` method are,

Parameter	Type	Description
-----------	------	-------------

Parameter	Type	Description
images	ImageModel	Specifies the options to insert image in spreadsheet.
range(optional)	string	Specifies the range in spreadsheet.

The available arguments in `ImageModel` are:

- `src`: Specifies the image source.
- `id`: Specifies the image element id.
- `height`: Specifies the height of the image.
- `width`: Specifies the width of the image.
- `top`: Specifies the top position of the image.
- `left`: Specifies the left side of the image.

In a spreadsheet, you can add many types of image files, including IMAGE, JPG, PNG, GIF, and JPEG files.

Delete Image

- If you want to delete the image, just select the image, and then press the Delete key.
- Use the `deleteImage()` method programmatically.

The available parameters in `deleteImage()` method are,

Parameter	Type	Description
id	string	Specifies the id of the image element to be deleted.
range(optional)	string	Specifies the range in spreadsheet.

Image Customization

Image feature allows you to view and insert an image in a spreadsheet, and you can change the height and width of the image by resizing and moving it to another position.

Height and Width

- You can change the height and width of the image by resizing.
- Use the `height` and `width` property in the `insertImage()` method programmatically.

Top and Left

- You can change the position of the image by drag and drop.
- Use the `top` and `left` property in the `insertImage()` method programmatically.

INDEX.TS

```
import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheet: SheetModel[] = [
  {
```

```
name: 'Employee Deatils',
showGridLines: false,
rows: [
  {
    index: 1,
    cells: [
      { index: 1, value: 'Mark' }
    ], height: 30
  }, {
    index: 2,
    height: 40,
    cells: [
      { index: 2, value: 'Id', style: { verticalAlign:
'bottom' } },
      { index: 3, value: ': 1001', style: { verticalAlign:
'bottom' } }
    ]
  }, {
    index: 3,
    cells: [
      { index: 1, image: [{src:
'data:image/png;base64,iVBORw0KGgoAAAANSUHEUgAAAOsAAADSCAYAAACrfzewAAAAGXRFW
HRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAAD1ORVh0QUxUVGFuAFBvcnRyYWl0IG9mI
GEgeW91bmcmcYnVzaW5lc3NtYW4gc3RhbmRpbmcgY29tZm9ydGFibHkuIDt0SWEAAAAVdEVYdENvc
HlyaWdodABVbnNwZWNPZmllZlRlZm9uUAAABGAVRYdERlc2NyaXB0aW9uAAAAAABQb3J0cmFpdCBvZ
iBhIHlvdW5nIGJlc2luZXNzbWFnIHNOYW5kaW5nIGNvbWZvcnRhYmx5LiBae8yiAAAAGWlUWHRDb
3B5cmllnaHQAFAAFVuc3BlY2lmaWVkozTnUQAAs/pJREFUeNrsVQmQpOlZHvj89593ZlVlVXVXX
9MzrRlpDh1IYgUWEgYLIQKwWBlsrwPbBBActoNDDlgBfrBg13vZ5XRgmzA2axtY1hAYWFhsLiGB0
aBbmqt+nqyuM+//PvZ93i97wBuxEev1tDQj/19HRWVXZf6ZWfk93/Oez2vVdY1mNatZL/5lN3+CZ
jWrAWuzmtWsBqzNalYD1mYlq1kv0uU2f4JP7fX0b36/53Q3bdfD7L57XGdTOu6yq+GfqvKsqi8/
4u+p2z+Si+NZTXR4E+dden3f9TJ4tLZ2/L/xalnPr4brWZekRa+I/ZTZ3Mz8NvtYOP8FdRpliYn1
1N+9k4QrqzaXsrD/60XhD9x5Yv+0Wnz12zA2qx7sI7f+1P90/3r/2AVR29bLeajfLXstwZb/eF4D
yfxB2L/1jVMp3N0hwn0B114nQ78wMH44kX4pY3KqtDpbaBmlmkazeZFli9c13/c84OvuPy2b180f
+EGRM36L1zP/Mq3+5ODm78gQHxDLRXj1SrGbJEGSQqURY3xpbN45HVvxOTGTaR5ghvPPAvHLQhKA
e4WnP4QvW4H5/ZGsgkq2K6HOM2RphlqeXyyhm31h72n272tL7zyBd920vzFG7A26z9zPfl/fJs1P
d7/1jt3Dr/q4OadywhbsCoLVZZjvspwPJnK9wie28alhy7gjW/+bDz34cfr2drD/PZN5AJIOx7u
Cwwny+xtbOB+y6fQ7c/QmlZyOMYleWJ129hemcfYa911B/tfjRP48975B3fkzafQAPWZv1/WL/6r
s9/3XKR/Yskdy/Np7OgSEukAjrba6PIMyzTWpgxRlEUiLISLdvG5UfP43M++y344Ac+gjpPkUZTd
MMQSZYhXixRi0PbG/i4eH6M7TO7iISdHa+FoqzQG23ptWoL8MPOTRTJPxGW/a7mk2jA2qz/1/Wz7
3yzs4qc33Zd+9VpVvVn0yXKvEKrE2Jre4y6KiFYFQCuxJQtcXwyw+3DY7hejcUyxuWHz+PtX/BWv
P/xj+HGU1fx0COX4ZQebty4gc1egGWYyOPMCA9dPouWMKoYxfCCEGVZoi0+bibfa6sWEh+kTl09L
gB+84Nf+D9kzSfTgLVZf27983e+bSNKkt+x0XpUPzDHghAdOq1Q/uMijieaLa8EfFkWICTrrOYzZ
EUptwtcP5hhKWbyX/j0y3j0wdfjTz/6BK5efRpvfu1DuH59hm7HRTQ5hNXy8fDDZ7Ax2hWTeoVKG
HWZxHJtHzvnL2B5eiSmsSes66PX613tdDfedukt3/JE8wk1YG2WrN/47ne8qXTb/9J1vXO2fEYOE
8AKPAQCmCw6QiQ+al7FSFYF8rzEbDEV0OZYJSUqYd5FnOHwZI7YcmBXfb70i98goOzjPX/yFF5+6
QzmixzT4yOxbifCyEC35+CNb3wdotNbqMUMzsXEzqoAbuBjICA+Pb6Bnd09pGIX2/nyaLy99y2Pf
tn3/UzzSTVg/a96/f6Pfc13L+eLdzphN+i0Q4zGbThuIIXXCpEKuwo048WRgB0oUaKQj3E+XWCvZ
iiiCPNZiuduHWjwKa5cJBUQRxN8+X/3NnzoY0fi6gyj4TaefvI5rGYnaMkhMESiPHL1LB577DHsP
/GnqFxFyLuN2XyCs1ceRLpYCdF6AuoRJs3EPrlfLx97p2v+/L/9aebT6wB63916/Gf/Wbr90j2b
1ZZ+sagvXGMDs4IyuYCVB/wWji69SyS5QRB0AI/uzQuUMSx+JSu3I6ETRMsl0v5f4V2u4f9wykmy
wROKOarsOp4s8L5vZfhieuHeOchl+FP/uMz+PDHPojz4y2sqgKjfoDP/YuvQ1BUuPhkR+C0AuReg
N3ds7AFvHFSi3+8gN8aYnrrKoabg/lgy/Odb/6af9IA9h4v5zu/8zubv8KLZP3JP/1G6/pzz747m
q7elJalm2dCm0WK5WK02dEhbj37NCIXSdCkzeu38bt/TlOZ2L6+i0NAFXwIPawMJ6Pk5MINw+PB
NAONns9TKIaTwrTHp8s8MjDl5AsIhQtB+PRnj760edwspyJqSzXqCv4dYHzF86LmR0hShhHsnE4W
SDojNafbeLw6lUMtvfM6zo+CBzHftP+k++5cd9rPv9DzafYgPVTfn3sF7/Dmq1W7+4PN9/AAgXbE
mNXfEMC1lHZsNVGuzWALX5rQaYU47fTG8D3Brhx+xDz0yUKR0wluU8rFN9W7pGmBSJxRrO6RMdz5
MvH8STGYrXEw5d2MTm4jb37ruD0dIUb+7fhCXNmRY5ep4Xx1kB8YxeTo4UCeB5VcOQqAm1lzzmen8
```

IMQd/YPUcjh0061AzjBG+e33/8rey9/U1NacY9W03XzIlhXf/0HrWUUVxtl8YYsjQQkFnbPbKLTB
soqQSyAu318KubsXMSDu+MORru76IUu7HKO+85uYHT7GxkrmOZLXNs/gtvvYbC1gXa3j8PjBU6nc
2R5jNGgjY88fVsA08VqKX5vvMTFvT1sDjFuTE7tAPvHc1y/cwxXQO9128KgM3QGffGJhckP72C4c
w6LKcuJfUymJSyrhenJdGzVxa997Dd/uNV8ovdmNV03L4JV2PZ/GJ49+4ZBXgvyTPeLKFenM2G10
Y4EpFkOHMxmwmBbKCoBXOAJyASyR00VJnSE+cJ2C47jIY1W41sCp2K2dgdDVGL2joQNT6czJGJSb
2xsCVtXeN/Hb+Khc7s4uX4V/eEluUaJtLiXGPZR2h5u3JniwkZfrulimZbIj4R5+2MBcCav10XFY
95t4XgpprIcMEXuYLGcXW53B78tv/mM5lNtwPoptz76y9/zA3m8ehMZMRXgMapbxiIEwGhFDqbw
w6Wkxn8fgdpKSZslGD/QFiPJqv4mHMBZy4PKsTw7Xe7GPXbYrYWCJjSWUwR9jpYLBZa3FDGHm7eO
cIwsIV9Z7i8tYl0dYozulewt3ceTzx7Dflxgn07Y8xOTwSkY/i+J8/VldxunWeo5MuSbWOVlvzOV
TN7NV2ZqPE0wuaocM37/vfv/qpP+2+/rQk4NWD91F1P/eb//Bbb9f9ukdEknQozRQIAH92Nbfmew
fFWqKsputsdJGKWXqsVYmFF2w8wEXCvkh12bSv7rZyPdG4WiIsCAwHsroC822vBrSwMh0M8deMQm
50QyyTA0XwuJq+H/ZMTnN25gOXJbQV91aXwwhb2j5Z45ct2xFcOFMxJUSLodnB85wYuj8RvRkt+J
gcHzePeEGIQIFsl4sseI6/uD5K8+MF/92Pf7Be081Nf/LXf3/TLNmB9aa8//flvf3h1fPVfdnauB
EHbx8WHHhV7WEzRpZi9YuIKLDEa9bG1uyM/S4SxNgQMEzVvZ2Ia91q+mMhTAW8qzFpgazBCu5Vjv
lwiK2sczlJMFymGalzmYUetLk5WC7Q8F6elgzitcelgCd+6hQuDECd39vWgCD0B8+EpPO+CmMUZV
lktLFphNV9hvmJHjyPm+amY3V2w0yfPc5zI74b9DaHeQF4j00hFfzWf/NCzdybf+sQ3fPEzD1y58
IVf8vU/2rTb/ReuJhr8SVi/+1Nf+Vo7q3/r8GQynh3egtseIIummc8mwlqp+JoD+I6DZHaCZHKIA
LYP2xG/NLQx6ngYDgJ44p+2hQXFWRQz1dNC/rq2kGZiEPsOytws01iWmMpTYdJQOGrJNRdxJKCS3
1cFFvMMZ8Ybwt65XK+FWbJCKWVotTrY3RbGzOR6qTD2/iFy8Y9dJ0Cv18aJHBbiImO2WOLa9ROM5
PW6bil+rC/XdcSELzCbz93r12/1w1b70mjQ+8qnP/BbX3D74+/5N/c99t1FswMasL4klU//2FffX
1XVbyzixZ198RgjQmBr0QJpAiznOW7dmmD/1nWxeSyty12sxE8UxrpzbR+3j09wuD/BVMzXduiiJ
2Ztf9hG4ISohD9dgadd1/DF9E3zFB1h0zovxM9kF04OX+4TCRtmwsTMxdq2g9AuBYQVqKCTtFUn
vZ2t4TVu7j2zDPaQXN6MkE14N7Z2YJLQ3zUCI7cEFzjudv7GPQ8ef1iC1eV+Nhy8KQRjk6m4J0Hg
z6mp6ed08Pppdr2v/raR9490Hr2P/6HC6/4C81m+M9cTQXTJ3C97+e/c5CvZo+Xln355PZNDMZjp
DOCKUYRz5QZmVNdzieYzGbY3D2H2ekcz159BqONTYibiTqFmL0pyqzUIJMtAEwFSLblCmE6ci3IA
ZCIaRoh158lCVveXLGCTJuQthws4xh+K0SS5mg50cbDFlrdEVj/kAoCd89dwP6xsOlygU4Y4AYBK
ex56b5zKASQ85MVNrf7ODhNcfXmdVy+uIvldIKHHnoZ2p1AWfhATOmWvObjoyM5CFI8cN8ZvPxVn
47Do3nabvtPnj1/319649u/9qDZFY3P+qJbT/z691nx9PD3b9189nJ/cwe9jV1c+/AH8Kbs4HxmY
UnAJhXKtMRCfNG2HyLav40zF3ew2XOM+zdvCZhqCBxRlZb4iAMNCNm2ilZdivmai/9awPPbaA8Dn
B0PsVhk2myeMoLrOvJpO5gu5+gFHQGUhTpLxFzOBcRtOHmpihH3338Fd25ew/LouIDAeb90Jc8hp
FLE4rvWqOIV2nJoHO2f4Fj84/vPbsIu5YRwG6Wg0cYdRHKxaxf4HSxUvB2O0N0e1uYTg6QrZbBw
Y3l0r888+fEPzWenX/EFf+sf/mqzOxpmfVGT9//Lb/6Fk+PFOyYnRwJE9/TaLraGfXDJBu5tnUUmD
LgSgJQCRFl8xCpfIRQzd7ZIMReW7ThMzzji16Yqu6Jtci57TwVklzPDeD4ngZ8LPFT7fLPRB0y2
qt2C5H4pKxgXEaxAL9Civdl4f/OlgBbu2cUzIe3b2E2maIUBk7kMLAsSx6fYndrLOzbUskYgl8cY
1Se+MVViePTEwx3dmAlmfqDDpYZD4eOppPGoz4u3ncB8XKFqMiIeq0xptmOyppfeuD+/+mLv/aHv
7fZIQ1YXxTrqX/3vW+ZTQ5+ZXY4CeaysdvdDbQ8W0xR8enGe0jiHFU6U6mVdHogJq4AsnIxOThCv
EpkW/vwB2NnNTEtHUuYtQ1C23PR9jqwGu11YdkUUQtLFemS/lgC60rLitbQFeothLb7LJKuFn8U
gKplM+elb9yDqgZ04t1bbXjSuU5srLS1JB11ep7iq0tZnWq+V09BOS+sZwabhDIAVJqZnQx18LKJ
18sg9HWCP3+UNUrev1QAJuImW3hdHqCcBCi5bbr2xnMz++ee+dbv/L7mrxsA9ZP7rr+h/+MluuTe
bS8TBMWUxQJvRRJypeRiZMVvVv+1921p4QNMTAqoiIgu1Ndob2gIcq9D2uDAMEbYD9Vld1zdN4
YxHCTOyha7U5nHhrqJQH5j443XJlLYwYikALARwhbA5Ga5MxGwVBuUhwZyqGNMCxFyDU7ka1PUCN
YV5/VzAmqtWk7xuAl0YfSGWAEsl/HZHbsfY2hjJ6+vADhx9PvrJq2iCC+f20GrLoWK38eyl4ijH
Ifinz/46MPz17/hL37da9/6t3+u2TGNz/pJW7c+/O6fWS6iyyhjBH4H1TBdt9Nj4FTLCeMoUpEyI
S8BBzQgxN+Refu9noCwFCZuiVkbaw7UcwkeH05gC601FUGsN2Qu1RL4ZOJb2gKO2q01guvw/qWLW
gDuC4hrDTPx921huVjYN0MRpnpoFL48L01mvyXAFp+2Fka2HTGJbQ1QVQXN2wC1PGcSjPrPrS0Pg
2FPCyOsThcbNK2zSCPMY91KbicYboyxMx5jKe/t9sExzo83cP+1HTmgOhgf3MLHP/ihvhv4P3Hzu
Y8c/OWv+cH/q9k1DbN+4ln117+7fe36jeeWR3fg6eREfb0zZx9ANDvE8Z0DbI66AlTx4VFZ7M5o
mWs5u3m5hgdMZhd2kKvI2auANelMew5FOUWJnU0YOSIz2q5tpq3tQCzyJbyVQpI1SPUfteayBcmt
ARURE1qKkY4kz8SgNUU/Jb/CZOmKwEs00QuCna002/2PO360VM9EB80TfSapfE3kYXZTWE1MLIsQ
IwzMch1IKGJTfZdLiPM5TDqy3sYbg1xZm8Xi2ksrL3ASMzqjhwO4XCE/dtTPpMTXQGRZv3PftKh
/7aN/0vq2b3NGD9hK4P/Nw7f2dysP8mT9AUJTU6smkFAQI8oda6w+RwH7bYqKuTifirgh4BQ19My
F5godPvC0g9BAIEulzKbd+AJhTzUkDiOMKO/F6L2VskmmXNk7kWQVCWlGas0K4wo6WFEXcWesQ18
66VYeJS/FsW5cMKFISZmOcZwS3sWzKPY1WocgEeJdQERaxspdFC2dZiHphNW14bsZj3Jf3egjKnX
ZRiky/kEGIQaraKsJBDIJAD58zuNrqnviqvv4uXsjrhZjNAmQ7GOHq9UO8//EPveef/d71phGgA
esnb139tXf1ju7cfiadz8dVnNwhIIjH/HkGO1Q/M2whXwWwRKztczFRBbTtSWs2m154td5mgf1q
bvkhWAYyPaMmj59RHVQawd3PzurFACKuZnHwqx1Jj6qQFfAXzilgrJe6wor0wrgSjFTa/qthQFuR
dwJSzMQxIBULWYv87j8R5CXVS6vdyA+bixgjbR2wg3bch+BsbyeQvzjWA4ayxfTWg6mWA6MsNPG6

dGR5n6PFgnmwrwsddwc9LBzflNeq/i/0VRNfAbGHDHbxbztbuHwSg9PJ5//TT/+W/+h2UWNz/oJW
fOT/Z+38mR89NzTApIUy8kSq1mM8XgbZx97JzaHd9AR5ownU+xevh9uOkdbfMEgdOB1u8qcHlnOZ
fBoAKc1MKDyffVvLdujbarsKDtf7i8gc9gNU8JhRFi+PDGNS0EWEZSAEBpFaSdyuxaTuBJT2NGgE
rWWFJAapBKflswppOnaofrYTiEHg7A3U7V14CroLddVU9sucn181+8jErBbYilUWYFITOkN8VOjx
RJdVxhcLjidzcRbdhFFN3Dm3I48R4DjwylaHQ9nzuxpwG17qxecPbv9k3Lph5pd1DDrvWfV3/gBJ
4/nzz37+B+cW05OkYgpOT2YYDzs4uL9F9HuJ8AQ7er60xjuPQA/LDWPykJ6v90T01L8UktAJ0x10
T5smqlUjhCTl6iTrkX+qQCZitfqknJv1bWBzNdQ5alymEhDMpVrc9kMjJEJy0LI9XCQgdGhOWZN
GpcMnpMVhVmpn/Kw6BYp2lo+rJsUV6UXEH8W8vV4gzuH/qoFGmr2m0dwbGKhYH18Bhf2MN8wmaDG
ofHU+SfHsiP2FGPwLdx9twenEoYt9fHaiXMLMx79uJ5jIab6eaZvTc9/NZv+KNmNzXMek9XNDt6y
2p2PPbEv/SCWMO8586IvxaUCJ0cgVMhPTrA5u4WOpsj+GWivaF+KD4mTWSCVUBBhrPJoAIwi2kdY
Vn6pBodEqa0S6rl9xhJkvsJMxKADA4J00F8WccLFmJUP2RKx+M1GVxizEh8zj1DFCB2wbQQ1uY11
R/kyBDWL3hfpzmzkGjSbeV/mXCthZDK4K4dJHC01+uyIOVsvp3BrT30++TLF6eEBRhvbiE+PMJb3e
XByir4TalEGK6+W85VGvct6hW6PcqcjeQ++HsD1iwy7/pYs5v+bDWyLvdgZdHiXbZtB57nqILDe
HuEjS0Pg15HfFHxPdNCNuYQw/FZbXVrt30xf9vw2n0BbR920FaT1hZwsYJIA0U0g+Q2VQ5rDTZVa
n5CGddRs5i/s/xQ/NuOmKm+YFZui8/p2CYgZckhYLTm+zhaRWQKKVytgKpdgtHX3/P5XAGV5nBp7
trEMOuPK30NLJTgl9ZUBb7Kk3pMKzmsomJreqVm93IRYSVsPN7agGeL9RC4WnG1MeoJs7K4I8XB0
cT4u5api66LFVJhe7/d3f3Ib/zEqNlNdbPes/Xsv/9pxy7TnXgxEVYrVXTMEdoKepvoCRMxIuzI5
nRtH63RBgLWEHmbusuEFaPQR5b41Uy0V86YCUBsnP69UFV9uqG1qkQbJusKOFX9cGO1gjdIyIMV0i
t8RQmcRv6WC38zFsoSwZrRYQMm2OHEZ4eamY40pGoqwab5X71t5jqodMsZEYCqb83SX56aAmv6qY
AGkPGERLWbzSg4R+V+aaRTbFVN7Oo2xcE9Rb7TV7N3c7CE/ZvljIn8H+Vu0Heyd29WDP78xhO/Jg
dXpacnlycGd8WQ2fdfdWn9pdlUD1nuy4tkzF+J41WdhPX3Hdn8sTNdDtx0KiwqzVhHY/+KK2Sj41
N9t6GApLbQnGMX3r01Ka2stu6v6vxarmwhYYUGhQibyqq1JjG3C5sSMPI71gwz+BO04AhQKwJMS
wYzAVJLwFdoiiodhCgKVdjCtZov+sbBtLf/nczHYdNccJiOT8UC9RLIfTWcq/NdkT0t9XHYNYc38f
B8uKyTA/tsWTK/nKpMaBLX21+6MO+LT+vpYX6yOd18AGsiLkefsjzFfKhFmTS0tDNnZ2f7cZkc1Z
vA9W8Vq91201+37QReV00VSeuKLDgSsPTF3ZWOKyel0QtmgBTjtDmRHKyNVjr8+Phnl7cumb5kca
e0oElpeS6uHjMMJbY0jMMmoHv1SoToNFhJIntvlfDWNazV/qazvmWsLuGxeQ/xgW8xkzq4hoyvQ5
KsSELHmWM1mmt0Ege3qbTK+s76v+r6WqjEJrkvlYT1hd1ZU8XriAyAU05qjPhZRJP5pJAC3MZsuk
OcpEmoSxymOnruNo2efRRJliJcn8LtdhD3xXeXAOTk+7PzTf/hXnGZXNcx6b8LrcF5Visk6W1ISx
cXuxh780kZ7OFbTEGLmscDB0ZoCAQZV0dz2uvEbWoigPqTtqy+qaRSatRoBEojQIQ4Lkz4RgBAAP
Z2uMVyp1rD+XMND9GULFFpWosW6MF9t50y7e076oHXto3S0Onlt81ZqxtPcJn271Y2cT87nsNkfK
0zJHlq5D306Fa8hh46ViWluV4aRmUMua402V/I+CwFpSw6pilVXDFSJCR7HMqZnLooZPMZikeL04
ONaV1y5A7Q6nX6vJ3844E6zxsxqvwuDr4HDaycs2jhcVdnY3ZUP04fV7yoJax2tTgLTSc9e2Q1YXK
ADtNWNy3KJdGXOSJqajBf68f2WYk+kVAasGnpRRy7WfSbNVwGPl8vtKywGpcghjNBuWpK8qPnJts
yoKcJiuqY1PWmnlkmFma82ayqJ3o8laUWzr9Up9kPyPKR45RDTAxf0Auod+iMApdfxkp9PCahZhF
rF6sCNgl8c4K2FuV306rcEIVn+Ak9lCGPDUTXDmks92N9DvsZd2EYin/Tny4KbAvwHrC7t+4wf+tt
pPk8NPkFJujbbRaQ6TzCcJOH36rRyNZWC3XTWm1XGVPCvDSH61L06PKnCqZy5i0tqntJd8JK7P+1
+CkWr8iff02qIzOYv6y0hY5vd8a+mreKuBLYyorqgpToM/UD69NNmb6BbWauPx8rq+BeVzHpGkUk
Lwi72tXpqeWbXrMs9KCv8vyws5OUWlnULRciPnf0TxxzMAT2FMbwfNN51BeThG0+3KI2EiFnalGw
drmyeQUeRYzuh1YRfWLDVgbsL7wwaUk77Q63SCPlgiENE8P9rG7t4uwK74pm8HddUTVdZ73Aa2aB
Qqe51BpddJLrNW8tLSMUEFZrs1jPkZzq5YBER8rq2o4WJjYMkxM01qY22JahiBms3euFb8G+OWap
Wlea2Co5uVNhVR9d2NYyPj4dceNTkBNBFmB/+fMfprRatSCQSjXvCf1150UnrzPQg6BtvjohYCU1
wje9z44mQmIK4w3PRzvH8AXk5pCa0cnSy1JpO+730jg704Y7Za31+ysJsD0wv8xHZxfLiaBK+yyF
P8riiL0ugNNgdhrs5FJS7OhTX5Tc6g0NS2TdhGHVoFk07wLSzL3yCiusmZhghk76ZLXW/LJh/e7Hq
PzJIGYn0OvVAoBSdcyqtS8qP5NDoyqNWLe5LtSn5aX+rJqtUnPc+LOuif7S12WBxvNvVolzE31e+
718BfafZ3SyNNv0PBNV5vXZ0cPUzu3TFY5nS218ZzeS59uYnKxQ5rb49xu4c3CC2zfuaCCqWQ1YX
/CVxNmF46M74pi2cXR0gr5sOqvOtdhAfUzZuJZd6+anOUqQ8rtubJJalekUOJbzEaTq165NX1Spm
tC2sJOyNDtp6kQZVwNBPATWwNH/q3dZqI9LM5blhxVLCHk5K1GwNLGkqmEiJm2q4OXiYzW9w/PDd
tbXLjRXZCIKoxylh9m89QGqD57a8nCDoSu2loWqRaCPI6NBaxVJoOn8hp6/RYG/RAH0wU8ObSy1
JQtntj3DlmeYDmfYVfciP07xzg6nfjNzmrA+loKv+elszKojNm7HwniBW5lyPfqZtqkQIQmSCiZtb
TUzK6yzE/RlCV4GfKDYDgoiuy40r2mpnZyrr8qmcQPoDBZ7UqvcBJkqcxgY4J1AEKJPJ/Dnrhym/w
gHLGoAQDKgLBnaoRCr/lwNE+2ANYKHBpPo/2TJkdGtdJPE80/NwcB31jTWAJH5AVcR6/yJLhNQDV
VmkfjFLIvuczSN/m+sHhxhv9BFHKfIoRtBp6yyeQn7XH44xn2dZs7MasL7gq6iq+XhUms6X6GgdL
RQg6jfw1frPba+LHEwrmqZvbCPpolKhBA0XR1Lz0w617UyL5+kH+i1TzWDbz4OeZqZ1NxnZmXwrq
rVparkmUmyZ31NNCVHL1Paen3kry36+TU5NYdyNDGMNWE3w6GHBi2mk2VpHp0sTyGIQiqDma/Y45
DlZahmjz9tsMKDbTb9WQMr3TOXDPEkUsAR2FC/FN22JRYDMXBrNi8ViqamdoN0MpWvAeg+WbMQZS
/BS2Xy65cQ3y6K1yiD9Q5qYhRb1CfgcZTj107nhafqS0Zjaod9KELH21w3MdZGtKwKYhRG2EejWV
jlhrHoNYGU75kUZWbY82CxNpL/phNoIoIUQTNUoLKmn3201X19fx11GVfeTzrpXTRW8bKkr1oGpq

qiNOqEcBlrPyy/9uZjUaarBKVfYlJIYtgavasOulaOPCcm2stKY7qH46RPx7VlHzZJHpn4mpzNc3
z/Fmk5QpA2xNtHgFWx741n840tTKdTrGadNENhRhphzJnCWPimiKEWjtZbMtam6u+Aq1em8a2K
qC1FLR2Fa6LIBjgSU0+NaVZXrqfkiDSgn9X/U2NImuAKYUTdlHGkXlOdt0QVJRqYcG+RpUzTb84H
pUkHBXxrtTnJUqhLI+cdcXrPK4WZJj64bK+2wS0NoVpwjNwxuCTU2pQqSVmrTVJkDKnFdZrcx3aX
EAwZnmNQDBMFmb7nJlFmuaZrMQ/LxPs7D7YbXZWA9YX3kwp8VxaIs3zNGCDD8m8I3OUjqvstLZGh
YGoUWiZnlQ+0PGf75xR8DKaK0xKlQYW9GvFD/1a9VVjUxRBM5i6vgw++fJ7YWayaeUEE8VAQwFU8I
CoT0a2Dgfi2Ky2GYIcNkVgyGMQGcwGIVXkajLLVbi5M5JlPSb9WLYJSI81Wve7S0dk6plBCSw/hP
D8WkoeVrdfytHKK0V/LRKy0qqpi0E1eayZnSkvM3/lypjKmPLAOpqcIggB+nOfkkg1wo871D7/Hu
/DIG/LGDG7WC7YsL7xNBlsIY5A5okWENMJjdW6KDZhCYWECZVg0mks27KhiAs3DWM7DH8IK+oAnt
70WyrBv/FYmbtmf6rXhBF35WVfVI2q5DW0EaGmxvsXbNKXlWioDSilhBnXYtbtqow578r0D12lpA
wH0MX01lWu237FVzg31MIBGq01Aqa5sDVSZwJcJOqmKIq2BtSmv2k61Ke7gZAAFOn12z0JHQJkKA
EcDeX9yClBbqpD7ruTvM6fekzyqFXAYdAnfC7QRn4x9cnLSn995atzsrgasL+hq9YZTL/DTQPxBF
u1HSYH9/X0tatCILtWCFABhio1WKZ8jxHYgr6nMq9rNH8d46/asnEpDapsK76e7GS5LYD0Uqqqbp
L2rBLGYzAwoMXhEUFcsA2R/Kx1Qb/18ch0W79dMrXQob0ZpfTCft+TcHCfQQcmmespjttSkmdy1P
0ybl4dMYor7qaJrPVBdDdlRD/XrdiWV2tFkqVWvoo0FNOWiVXf1TY5pnqWSSpsb6lg+Hy1NK13Y
ppTS2ogJvRyFgVHT69eanZXYwa/oKvdHSXR6WGqKQzxOa5pBU7VBD010x1jMmrABBGY2uYlu/5K
jtak01VBeL/ceF12Z+CiPchKGhK07xWhmYetFJJF/qfJpZbaSdPrSauq2Y3/dWa/a6MBivrWUR2e
WbzNTGqy5RQsbY42XYnZrXNEZHiH9vymIqlyRrBni9Sre676R4+N0FbZdrzqqY4AWoz/1rr6A2+X
t8KtNWuqii96qNsC6OK+TyZxFr1xJpkzqj1WraYyT0c33mmqWJqmPWFXX/1W39YCMpdMcdXfoki
5Em4utVJkp7t2pJI71iouZppvUO7EopqJSvOVf5K12T9yQA1hVMFRXyEzhraA1/ZdQHfZYMVhkgV
sKcPBCokaQdpPQbyZLaeSM/810Fa81Zqowui3laesYEJvNCTGUVYmP8uLQ0VcMvvm52AFxRlcOi
h1Ib4kMDeEtfsI720EOFOePS5HzL3IyTpD+eyTnAie25uAKuMHacVGpZTGaJMG+lxM2JBLG8NdYRM
y874/jIzjXM+kIv2d9LyqssV5GqA3JjctZMO/A0faKtZILQbDZR4bPCq5ThsjxCcXjT6Ckp+9YaT
aXMC1MsjgDD6Y7UF1T/WBvASwVtncZwvJbpXaXaYgm6emqX/T2FppHyxUxV+jMxL5nvzMXUTgUMe
VabiLCYxnkiZmiSaA2xoy1tGVJ5HC0FT9NALprkWom1zsfWMM3xrFwq6kKB7ZbQSiT10VlwIddhA
Drk4KzSjPQotY3PuiZl4Gn/dImNXkeLPhI5xOaLBL3toQ5zDhzrVrOzGrC+8H9Q2/43wq6PpFkVZ
Mul2MaBlr6WTTdULMmmnd58WnyyWoM/QoeYnk6wiiNQDCJOBVCUWmEUVCxECoNN4fYOHdFCHI7
e0LApY2ydlqGsVSv1RZNqEJG6npyRRJVvH401ZRs3g5U7kUDp7KxFy+feOqHCicIperEr8tjNsK5
HBZztUE3djsY9Traamj4wtHL1bqs/YGfThT28bKxZ8pH6qKPxvrs8oo9pP0eXCoNrExlwOPFoS8T
Dko5quVStxQ54k+albwdeVotzrCqLGyLv3ZsmyneVmdNjurAesLv/zWP4WX/H3B0phiYrFsFDIrg
y40dVh11XVj3NtCvHd5tMj7XNtt8Za+RTFc51gzjEafleYJVmpYj4DMpUvPj7JbYSjXfjKpH+us
J7+ITt4yIDMt1pmHCPByRGSy7jQ6qWy9BCTsS5MCL1LF90KSTJDTqrl+QVAcjjcd+khnZR+49YhF
skcw56HXuBpoCk5OUW3m8mB0VoPxjKdPZVaaqm9MT0zep1wYZtqpZqUxvshj3MxY+P5VDJK0+HP
qsvJodNnifP+7xU+GcLXSLvubacVN7DfrOxGrC+4Ct0wsmyxkI28ZjscCpMpWMr5HeLySGOnnkCv
d2Xo70xh97WrjFp2QXD1IrY0OlqidnhHcwPbynj+Ny0nHMjgLNdmQ9nc5UntYYbykrMoVZ2qfpJK
rhtJVo2yMcybZSkC6TCZE7oCmN6qswW8zVHLWCLjx7hd2dswhabYzPnFXVRZff94spZqdn9LCh8
He8OMViOZXnPxWwJ2jJ8w+6woyBKbSo67W2E31ciqrJz1k66NqZF1TkBOFYpXW/mRxePqPaBY3pV
JjZFeujXOs7iWUhr7cdCNN7Pf15kgqYl83OasD6gq+/8T/+s/qHvvrzDwQKl92ixqA/guWx0dtFd
LSPzb2XYXj5FVoo4Lgsqre1ZJBVS2QWP2yhNbqI3Ys7SJYLM91NzOksilFGU/19V1vcNPhEv5Ump
oC01GYBqKYv1fMJIA6bUnEJMSspE5pWC238Ho83EXS68MXs7Q4vot8bod1tmz5bWraUZgl6GLQ98
VlXynDV8LyAflld1k2Ynx+LzRlityjV/KQZXR6uYLJWBow6TvK8404FauZjlacUG9Fi1jAlKlhhHn
IHj+8LiprmApiylm8SV0MfXJmi1+qof/Hdls7MasN6T1e8M/17ghr99uH+t3+n0NPdJP3K0e1EYa
UtM3MIEh8RcZN5TRRy0n5QpENMWlwsgbaeEK8Bz2mxB68iVe2paOr4pbmf7ndYQsTopj7Vggaka+
pnaKM70iPikpfiopvHbR6vlaI1xd7PWua9UGGTfKev1B0dFZUxXDRUsQg+B1Rar3hQ8hCwNdAv02
jtYRQtMJ8da96t1xgAr3PJlgoxulrCqJIxAthETN+VfPUzSRZAp3IgJGLiO+2hmMXJ8yMlQ6+vY
0RcFV+z0PJbTXFwA9Z7tzqDrT9tT2/Oq43Nvh8GWqzAQVRVRbHt0ohl26YUjyMu2DJGny6lnEmSG
zYVPzW+c0vM4plKg/Y3B+hvbIrZ2UXN8j0BsSWmoq15U1dLEAL+9VWTCBmVGTjJPDN+YVWaua9Ka
/QTBZgc9pgsQ2HZjhYwOL4DThHQFjoeApp+LdUvzeNigz4dRrAJ/HakwPd1viwDVzTJTWueAN5nW
WGtLXE8XJhfzVI5gNajOFbim1MBkQOt2LyeC3B5X0a/3drGuVaASE6asCfs3u82keAgrPdu/fXv/
NHyd777K95rxYt3ZIGwl99WFUNOKFclQ8voG1XCeNnpIRYHh8JiY1lib54Egw/GtG3jm2rM4vLUPh
72fgp3ek/tyCDi4+Oij6GohhK0leXVoChq0cohqgvMj5LJnZLmrQErEFF0tZwIMMqCjOdt4dh+hs
GN/74r4ziMxT12NImdHEyC+CZuHRbwQ0AXoDQMdhux0B/KcpmiBBRYWSyoJLgF4LeytGsIgg2haA
FHm60ixGYLFx5Xynr115FiLNKp6XTNtyUuz0BO/OU8T3Lc9MLIy/d5cyPzvNzuqAes9XZc2B19hZ
+M33YzzMUdo2BxJ4VqmNJDleeK/pbefxoEzt9G98hosV1M8+0e/jj/6wNP4WDRE99Kr8Ht/+D6Mt
vq4b2sXrwtiPCIG4fU//mOcf+1rhAF74sPONcBTCwvb0SmqxQnKkzssoxa9M40rMyxTz2UquzZGRF
pbTE9y+Psfe2UtoDzq4cXwNUZThg8/cQNEb48Yz78PZURcPjhw8sjuEn/s4eWKCKnkCFx59EMotL
e080FpgP9TUjK9T7tQmV23imlpNDHWzp1ZY1nOMIJvvMohUmWl39F+zGC3HU7Zm/tWxCGRLGwOYm

w1DF5fP78z3o/yJZjc1YL2na3dn/E2R0+nPdm6paUm1B21hE/PTUpWWBU6evYrMbePd//ZnMUKdX
HrT2/HoF34hkoMFWhsPIpnd9+Kpa9fwps/7Uoycqficc5RPvc+My2DXjTAzlhPTCBAtUS0XKOX/+
ZKC2bH41SWmSzE/5YBYUt+o8LC50RHfcopcrv/6t34ZThfCbrcjPP3sBL/zS7+Mhx57BIvwIp74k
8fxuu0Ozo1HsAfn8MxHn8bOmWpsnD+vHTYlFwqqQoh5b6W1qirSBC7K4vkItapW1EYnzhWWThZHA
sptVfY3LYK18XPFz091Oyx+wBk5BEKfg7kEsMkqe9X99xfNbmraes9W/Yf/2gqRf/mZM/cF+9M7p
oMEsWk2Fz+ttXC2wuvpRLE4j3L79NAZ+F3ubI1w6fxbhA6/CH/zzX1K1v0cfeQjnL34uqniF85f7G
Nohjm/sYPax5zC+chk2e8zWA5MhZis46DirldQtFgvxXYVdBbC5tULOpvjpkUt/0X7tG/CKz3kre
ruckbrCwfs/KuYnh2Vt4+yVV2Jw7iGczXaxOLmK937gTzDqOdjptDE9mcITP7k7HGnkmC5ona+M/
KnWE5suHR23sVaiYM2zJ+zJlgFPm+rF0m+1VD2CE+FZK8w2917oYStoYVPAT62nthui2D/tDax6Q
y5z0uyqBqz3Z1VVR+y8HuxUWEXYxRd/1W1g6VzAFSA/uQW/NcL1N96P+xiUSUskwqz1ZIBsA7+Nv
7FX4kZ0B488tCEMFICe+KGDp2pE02NsJPzw5uFPE58xkksRHB0d/VhqY7mZxVpbRta0EnOYxbiZi
JRdKY+eu4MxpwEsVlhc/7gGfzqdHQzyCNMqwtD9y7fi9p1jJNcex5U9MdfTmUqkTk6XSIWpz9Vd+
ELrtTx3Z2sEry9+rLb1qRVatja9c54s576atrpCvzi73eNsG46XpLiaTznTW47aNmVgNfHdqeFv
W4LroC3x7Y++ZtU01X/aHH4d64A39Vsqqas92RZjttm+1vgpGgHoUZaFULUOKKiys55BP6mdtuUL
LAXbgmFJRkdLecR9vwWztQJiihAMSnli7q7S3gPPITelcfEHBaWWH6aAnjNVHA4FvAttmavrUxXgZ
o8GGSqkmLcANCLKPPKZr9fKJ7/OsHzfH60+/qze542Vi0e8GLOrH8ejVYXWZg5b/F+nb2MYbAsYB
0Y2xnfgkA3ltVariH0BRuBN3oKXZsjEnrDqdSE/c75sqJfDyrfMNL1xv68MSx83cBz0A08Y1EFP/
077BbADOQH63VDnxZYppyyfTQEYML0QD1gas92wVmW0VVeCENS6c24NNKROXozBCo7jPQchagkeY5
qYtzue4R/Hf+iHaoSVmrGx+29Ts2htb8kFtQ1xgOOUKXjHXyHLtr8XTtGF0XZRwtX+VQRsvRztNB
WAOFnCuzYejHjrDDQEx87VyoDBAJozbFnbD7sVGLoWQytmqNpQzpmMmq8v9syZV92JVqSpXOCn90
8KknlhVte4UstnOp5Fft8zmEC1vgaEwr6s1Eo4Gkgj8jXaAUF7XhXaIoVejGzjyesQjFrchWcnhJ
N9Dz21kXRqw3ksZ2MosH2mwsYlhbywbPINFGRQ2m1umWqeuY73NSK6ClHpMqqxfaleLMnAQGrkUF
jjYlpuIaw407yn3e0JILt6LYv9oeXCPDV9R3vdrdPKhfE89ScDYa9c7pNNbiORawfDLR15UdRyH
5edcT3UdlTMRdhZ5KxOhgKblfkkSwc04DKLXDx+EoyXXOVqfkFCbHSz03LO/+GSrN1YZi8sZyf
nCq3DLT3nNhVF8jzxd3N9EqU4xoAut1jcB5Ke9B5V2ymddsKLOaftZ7EWDqbyb15hiV10coYAVb0
GCUAnV2DTdzYR31yhICEiSnS7xvDQpy/JYSOH12mgPehj2LRWKcCjvwtGNLORflxxWqm/mmpRK7
awlQistVAGf1GHoax6z7dnYsDL46TEQHavsqRY+6PS6zEye055UaNluWa2b0hnNdRzV+aW/qa17j
mWmBFBreA3O2qrWsqrluuFcnl/Yv+u10Rf2bMkb4MHRYhWW3Gtva4jVnZsqQWqH4Xouj7xuYVSPJ
jflKuWa6a/+pNXsqoZZ7w1Yu4PUSkfisPbUv3NL0/9pfmm0g9Vslc0I3ey+So6xCF61VLhhhW2pr
sCRjSij9YhIo85ghS2Vf9FRkMjMVDot9y9N2aKcZNOcrsf6HIY+BV1ewtlc/qsYTaFe8Wqy6QzW
i0YUTZWQREdl10yLFqQnwXaNysvQwBU8EaeG61g1wh2QxUL5fWKOfH8SCZL1r53pmzcvQcra1oOv
l2i3Wlr3+ula0eojo9RrirkWzF6/S46HqVsxEx3o+KhuzeH35fFnrzSbqgHrPQowtQdVvQgye61gq
CM0WAhhRwdS6fBh0yCu3zk5nDkVFuaTFFn0Le9ptj7TJHWZGCLSVXpwjL5vbTR562ypEViL+RCVJ
HXMfHRCTcxfr/JNs7eYrSxS4H04R8crXTiLU1UxpAgbm+ZtMT0VospqUC1glvZyTo+tc1oFtDSp5
fWWYr46qlZYa9634DR2birKyfClUZzcrxC2WqjkeeJtjkIPg3YXWVpjtljCF30cJnayXMGX1xW25
E4C7tZohOVqAe/MEM5eA9YGrPcKrHuvqK2DD+UoVmpSejqkyTaCZESGbWmld2rZ1PGpafCmlKhOc
qtVzVAVIHScRq36vwRyxfSIqt/LtfJSFSeoGUxd4loAzhk2qiCjahSpajdVrCryxcSkJm+5nla+1
ksCr+GLWUtyXM1NdZLcl8qMLHioa107az+valip/jFHZtjMHWtt87qs8G7Em/50YJsxHcLYvJLnr
RAGnuoBt+VnoZgJ3XZbyzCz6QTFItVIOVVtcaZuk1i7VOC1d3u5di5WDe7qgHrvYkv3X7SEmbTC
gCXwSUV2A5UhZ9kSb0jzmVFlaw1gwtTkcQmMc9Rv5MORlq+9HfTI/9p66/Wc3DE57XySmVOHQl57
shGKikSzL4nQDKT4jjFJRPlwKYCmtvskmEUuvZUTI1yMGqkKyt7erCYca6Vmb70+10VKHeM6hN9V
duMxOADXctFvm421yZ4ywJGqaMp74cjMDgcmewfStBR4AoZiQOuAmOWABJvJLXWiBzU/WrzeDoC
p4wePu+B5uumwas9xKtJQtiFF5562qlCvVki1U1dGqjhe+/TNFoK0Ppz2rL+K62u07FGLPWUj8yV
ZDQuAuLvmJm9JrkvkUcoyzkKd223CdWRQOCTpE+rXQQsxuyf7ZnxLfZzE69RYKUDMmCfM/Rulwdt
qxtbnK/wgxPpiYUu4KIPmt9Pb40PRA4FiTrqlqPGgWxCr6sB0LTD/ZaHvp5ICbuQPxRH115eaeFi
4yHkJeoAFs+P9XHVME61qTM7MI7/+DvNRuqAeu9M4OzyNLJa6WtKhAsflB2Za2sHWiVkuVt2HM0C
KQTzBmQUcFSGF/v7rxU9XHFHxVLSU5XQmqseEqIZ6pgJF/RxyaZ301h8Yd6kUj1PSrpQaJvpF6ozh
K1a2MxMbyNjm5mrtoqW8f6aC7bN0CyrSLV5nKMZFHREv4CfmRpgdg4xWMX0DQXEYQS/qU5Bi8EOW
ipVy1Y9erE6sKrTEZ+0Da/bhy/XDpMC6SWuqkaLuqIs8i4wU6nooVhFFCHutI+/s5e9odlQD1nsXD
S5K26LagjVdTrZ3dU6MynlyYBTNTHafUFQXk8fV9PXMekQYD2MuFqPrjBqEWRYDq3SqiHVATbsy
YI11TyiKqGYkDRzHQGNaxlgJ3miQ5fTYiFg6KK3MRQT09EiBU3dWMy0pknOtAwfo9FfRodtoxVc8
WeZEffWQBhb4taT47QvlyYzudU183oqmuL6fkqNcOt1VUFCDi45FOYntxHNJnJQEPiunN4muItT
Z6Vush9zqL9GW9v6oIbsN5DKziZV9p+zWqgOjWsynrdu0OcdECra+a1lqY7hWWHhCALCZY1VjDq1
WkW52Ap2pcCJlv9WWhm2IShADmWH1vawG2tVR5YQlirGLcZgcE2NkaiWcEUzxba6035Y7jtttYUm
cCQPH+rszZ1LZ1NQ1+6QKJAtel/B20dMUKLnwCFI9f8XqsZvbYkmdJaP29pmQILBqvYu2q3Qo1CH
x0nWE1Zb7wy8284KY9DgrIU/U5f/27V7BS91539V81uasB6b1eRV3UWA1sXUTGXyT+zZabDqc/HW
17LsI5hXjE/aQpTEpTTYXVZKwRYd34jKkSpIGjZX+aRvEEOGebViI+sTCmK2ZkkVLiRUBH97KwV

L4FHLUYBhr5pYJwJqCJaX7q6A5hyvYAYWbm2+g4SUzi81wA2hYLINYm8NrONbdqq2w4hb4ZJfbU3
2agS4cmq7iEq2M7NIqsw7EYdXa0HNIiy2uTQYn5wSkqYfte6KLXCbGkwHe0QLcXaBEHLWE5fm4OX
vWZ/6jZTA1Y7+0f9TWfX9W/9kMro4JPVs30T20rg63FslXdyR1IKVP1R2sCbzET/3JlJq11emKud
pWB+cU5rVSOKMUndDojVbzH2sS2MwF4tBR2bAmjZardyy4YT56TqZrKNXnPqhRGo+L9nKxW6AHhD
LbhsTVNCy3kkWRZF1PQDHbNvDudoUwTHpmpdaZzTPJn+SG1fz12gwEym+LdvlEUZpCpMvNptceVZ
YwC+u2xg5EcNq1WgO54A7ndxnL/CIG4AOzMKQ/voDfqPd7/6/+giQQ3YP0E+K2tUUb20/LZ2mgTw
b77567VfLU0QCSgFBMWq4myZDE/Rh4L83SF8eyBMYc9YaewtX6kY/xEnfgyGY4qKD1gIeAiaomyhp
txLGRsf1DYawhaDT57Rd6qXefIoEmALQ0e8m4Aw8IxfyfQOWbMiK6cmkERWlwOfvuvdemAzSb3Wg
6Is1kOW+ZwVB2eF5iDCOjilnr5DQ79YyiOzHJ3NLoLWAFZ3KM8bwJuLRbEQ6+DgUADrXB0/9qp3N
LuoAesnBqxb53/PquJHGQ0uBUisFqolFwKNqDIXqkOLWXifiskrpm8+OUI6vYmq3UHQ34EdCqvmp
rCBYy0YPWYqxGYDQDTrA7CKqIqXGpwqWp5GX4tsgXytWQtffIfp8pHHe21XzMwT/T81R0vflqefq
V/LgcYaCCPA89hMXrWcdVpmPWjKWk+N4/zWNRgZvGLJoSZhqclEg1sVEo3/69QUdMtRimuQLacop
2JBJKH42Zb6u27XV3+7PD2A59ZpZzz+kc1v/Mm82UUNWD8hyy7K/76up18qt8ZaqM8aXu2QWZcIE
nTMnbIbR/y1YnqI5OAZFJ0Wwt2L8Hp9IeKWkQdlzXCcIT9ZKWT6g76qJLJgnxVE1HMqbA+FFSKKp
ohXKx2GtTo90aomipr5/aEKm62OD1S5wm+ZaK3DSgoBWrI61WBTZ/uippMcYdNSyXmZvsn0UKFEa
qF+aLWOpJU6V4emdGkmaZlgFS2JIjVlizraMlUrIpmttA9W5zXzfSfy90g46uMUBRZNXHkQrXT5+
JUf/uUfbXZQA9ZP2LJe/dYF3v+rvytO4Ds0/WLyMWaoU22qeyoOfhEftTg5RLR/FVkvO33+1Qh7m
xqV5SRyaz2e0aXf63ng/JxMy/FCBWuZJsgpBi5gyZIFksUCq+kE0WSpA2Y4prEWXzeZ3oLJzghAx
Q91r6mqE9q2KYZg4CmawZ3dQntwDqZ9vXy+7a3SAgtaA6let6jNYcF5O1owwcCzagfTdF7PmnXMa
2eNiAbWqIgoB4NjW1qGaKXy+OUMCDvCrh461+9/Ynuj/+Zm9zRg/cSbwjv3/VX79uI52dDnTOGvm
XrOIFPNulzWB6cRsukNRLMDtF/xBgRhX4Gqpqj/BZrMTJoesUdCANxYltp6gdyAYBWEZHRhH0du
R2KX2m3QxQc20Ef1a4FnK4mYxnpJfOxgMKKeGoNErGJHC0f0eIYQSCvITCC4qW22wlHOpXpEKJvy
shSYUoZy8JMjjOlS8XzZrGWMmpzOuJrM+1UMg0F06Prsl5Ynlcn4xX6PwzPXbi5+zlvf7N9+dVNU
KkB6yfbFD7zcFnMb06wuH5084xM+HOza6BmPQpxeoZVwT7cnctoiQnqaBuckYTXK+2vlRi4lqqh
6QupBtq3tOpfDNA2fc0SOUJQHLHvr3fenGCilrJ7VLzq/QN2WZXeZZeA6q9b2rzK4psK7gyDUrNT
m5iNL6oEV+Hmr/CljSJWcGkQ5qz1Y7rUF/aNk3RbM8rtRLLTBTgugl+W8DZkdckZnch/rc1gLU8R
sgRkXIgVGWY6G1hlmvNnYdf+7lnL7/6TrNrGrB+0lbl+v+nbOZHuaHZskYz0tJ+0wB2TF/1SPzMC
DuXHxM266jL56zVily6gGLC0pzUfCvNSgZ7KIDG4U9ppe3rlhbUV5pjpQpKLuY0MmHX3NZifHbRw
DHF/axwuqtEyABXqWAPxLSW68ap5lzzKKYmb3N86b0kAGlfKX3NTNZmb6xVGWClve19ubWKjF6t
5+VaSo/1RVzvQ+3PUApvvPRcx8U9zfGwTLFz/3hb817qmCJSb473nrq57722xt94Aasn9zl+LOfE
sx9tWXVfZbnGYkWX/3RIksQH95E96FPH7dxd2YnpKwtORPSw4JLgKGYmK2EioyB2cWIAyOUQ20
0KhW5etdyF/CWc1h8cIsyfmbpIqfZasLGJBvxuqD6t5WNYO81yglj67a+xAQFVq6mZ+cgf23obWG
lOJnw3wHLrMActkeaZgjX6LY3Rdm0JhAI30zQg2fWq2BbqtdSdOgVx84lJM8eu3D3Ayj8XclgOh3
8H4zAW/2SkNWD/5gSY/PCjK8th26j4tWkdnqpr2TEKQE8/HV14p9wsUDKzyoe+nTels3mkFGkV22
PsqjOgmppoInLfQD4GehTLNtAywFNMzIFOKH+gGG6gCAWWQKAuXdqXBKAwpE2gtL3Fm5RRzETBxq
h0nzngr+46pPFR0b60/c86kgKxyPaC5U109XI9S5mvGegKeFkXRZEeuYyMtLY2s9F+2SHSzMWA1E
Qb3fVeHkfdahQw3Bg1YG7C+CPzWnUdq+/ofrbTvUxjGWqc32OFC0LpdMRUD4c3kScuAIL4czVZ1+
xgtF1BW8oV4ohVI1nKh6Rg7rrUogv4v6ZHlep4qA7Z0lnP0nESLR8sCaY2Cwd9VYpgg7unmiuO9
sA62kkTGPCqOlIKx20jXUzg7Ih/7NTrmmA28BWmaorMz9Y/Y9xrt07JCIz2AF1GjYLN9lofzVLH+
bGavJwOV8hBxGZ0Ff8OfHmJjeh+A9YXC2Dr6tfqsnyU5Xq1YxQObbutPa2tUV9F0OKDJ7CazDC49
BrNXRYHApTUN8CmacruFVf83YxRWduoEGqRP9XwbWR3YuS0sCdZpJYAIq8FFI6KZXNwchlVOsWts
l1XLD8P2MvKlrqWfLpUppNWDgALXneEQF5nfHCEUiVSHA20KjVfXGtJMCuZSqZ9GG1mdFsVHqolK
G019bX9T8m21vSSbXraMRcLoS0Wg+/5+rNWF7uJADdgfZGYwrX9A8I0XyFkM6Zagsq6UBisNYTf6
2mNLpvCs6M7OFh+EPWqgCfbtZs6Q09VUzJlsoJRIHNVo0kL+WmSctNl8Ryl/DgVE9trd1GFXRSJS
ZWUXTbvLDW9wrJAFjfk8wiFb0YuVrZcKxADuiVmtl8jaIfae8rGAuZjdeodI9REGSPa60HHxd0mB
Oo2sWuIgafKdA3ZLIrW26p2oe2B4osXeayHEl8CUzauV2nwTCx97F081zSYN2B9cay6tXlqpceLE
uXYUVUFsYVOOFZDxdAKYc7BDvoXPRx95Cr++Jf+EMvaw3G6wETA+tfe9nnYee1jCFem6N8dnkMtG
LNYvne8QLmaa4G81x5R8AlFukSaLNj8o89R5gnSNFWt1B07TOVE8QpJnuF0PkGWZRhQ8rTrYOfKG
NNSttpjZcxVbU7mZWhMz6+aD9cBk9qZaplywVtaljLoF3VvxVa2wQykJY46zk4id0L0OqpUCDnIwD
PltrYRKk9nRycnJtza7pAhriyMi/NhfrvP3/UMDB851qAIKJVky2bwjOMKCZXQhtncXvjDs1kM+X
v+XHHZ4PR9HlBb40+eewrVOgMuvfwOCOEU1OYSzfQ7e1WuoOqGA6VXIrj6B6onnUAsj5r74f4GN9
CDCcj5FIOyc1svkk5UWKgT3X0Idh5i//204eTrDnzx7HdfjJT7rvvvxsp0hXv7pW8/7tZ3tcyZtI
z4yga8i5X5o0kf0UdeiT9RiqjV95Imf7KuiPvOrWsko7zNPI6MDRYZu9TEYDnEkblV2oI+ds4/97
je96yeXzS5pwPriYdcSf6+2qt+W3d83KRozC8YZXTSfDbLJ3VAYdmuIvTe+Cq2NH16/CvGWZ2/ja
L5CuRCzdnMbmZjM7a0zSFZH8v+LcNt92HtbCDbGYvKK77h3RqVc/I99CP2rz8F5+BXCpjHif/8Hc
C6dh//gK7B84sM4fe/jKpPd3IiWgr+O5mHvf+wSws2esj5zr35/Q86UCHlipqdrjznTNqXkWK8xr
wjidYGVBaN2YYV9FSG3tBDErr6Ya68sI94QtmYU2EaA7e3u7z76mke+tNkdDVhfBkbwnyI9ngsK+

ty0zLcy2GP1NwVMifqB2gDeEp9QNvbGIw8h/+BTGLzyZbgsjGr95m+j2N5AuS2m7s6DSMXkbQ/GR
s6lNUJ+7gwsAbGrQ8cT5NvbOHjmw7hw+VHYJ7dwbMfY3T2HZHKKw499GBMxQVeLKT7z7B5eeWkPF
15+FuMrOypoxqHIZEItN5wvdVwk28kp4k26rEoj5EafWaPETAixSELHgNCvDk1JYmXeEwNqXrcFd
AZ6jd3zezxQfuUbfbuRffXGzMXqvwuhW8Jp3lOm7f/wZx6rOsXtFixy40dvCQIVrCud1k1c6pyagZ
tr5lZi9S7h+B1YqQHjmnQwPPonsz1Sjtvnm0awMtBChEGC13JYOaa4L5kFbmJfyLMcnSD78AbRvC
EPu/yay1QL9lQoPCAU6/QH6D17C5uUx/G6oqRSaAASp5bWlGydp5siU+R1VntBRG3z1fM5aM0MoM
nkMix8o00P2pMi40zI+LvWjVuJ5t3roDncwizNceuDKv/iir/mOL292RQPWFy+7wvmbYkJ+sPasf
q06RaUKfzPJVEObA6VKS8vwWKzvXbwgDHSdEvioerZqLnni53oHU7QXS5QfehZZR1jw3HlVIyWHT
8ISsxbLGZznbuLSM9eQfvAnxReuMBA/MxdTmjnRmqR4Ax+t7b6Y1G0V5aa00/PRA5ZrCJvmsayFF
Cz2t7XHNdcKqVonnxemfnndZKC6SewUEoa1OVyZOV1h1+Tkthw+Lux2G3VviMXhR99nhd2/2eyGB
qvw6hV+5tdcy/7gx6/Jjn7U8n0TiaUig9dRitIISWX8WUtnZGG382dh3zw2RfwbsuHJSHLfKhC2n
c+ExFy4UaXpnfrgOip24UQz1IncZzBAQCehjmdkVA7EtFDRE+Z2+iH8LfEdKT9Kk5eMycooYWQ2j
6MSJmW0WfzVSpXe7LWWkouc0jLstsnXBRI0hClFyoZ3NpsLu9I8pmJhIq/Jbnl8p3vX905f/9nf
NaX/d1GYb8B64t/VZb1E3aZ/UiVVoG15qKjzMLCfXbZ20vADYpEfteFvTnSA4cnJpRGa5R12frm
QLSE3OYmknDkQZ+qISIO5mRM/VrLfYvOJeGLB46ci648He7yFiCyEBSZRhV5WPW+sYshGQ5YiGPz
Zny0WFXRqFQ/VQxe7Xpnf9niaLfUsa2nbYZ1dHqy20P2eIEabaE43R4IKR+b/A3PusL/lZTAPH/c
zUjHz/Rq7P9j4vSeq6oUtMPyolstTGBHRX/No3aOkFcTFvtGN8SIN5/Xr53tAXNctedMx6L92t1Z
EZzrTzWbhdGmSynMr3ubIGT5/HEjO6fHyI8t2nEtIVq2f4ml4DHIBCF3Sj3T1EzBSRZVb7qdfFDb
Q6RUkxifhVysnBkRsE6Y6s2SorUceRISv1FthJfd3GkQmzsuAmGO+99+We9/T3NBmja+tIxhV/1D
nEB3XeVeZWWRaJCZDqMSQlubQYzR6ItcSzmN2kTdmUK3hrCum8L9aaYmWPxVUCypGPui3gKlbad
8qJco7gpQ7N4/0zLXRefh7BhTOwaVb76wlv8sUZsB7HeAhrmhBXZUTGswz5aqVdQTR/c2XVSospq
E+c0z/mqB7mWlXQey0K7tHU7QmYYy2HLBanSODzeVxy1N+99Lbm02/M4Jfc6r/p635u/ns/9vVuU
b2BDd1sfaPmEaVTCBysmUz1jzT14WuxPmuJ0Wan8AD1sCVg6o1DGpneUmEvqxyo4j7SBB5/nvdhe
ZZqIdGXVTYW05aFRn6rZcCq0iuxaXmT3+uwKWF9FjLkearmLlUtaDKbemPzRYUllhIWJoytTfLt/
lhTNVXOxoKZnEmr5IslNh5+9fvu/6y3R80n34D1JbnEtHxzXlUf86rqsiWggJfq5mfAhsEmBphM8
UGuMqI6/KkyI8bzj1qpFrCnpX0c1liHXTVjjeIglfOHKqRNMDNCzOJ9h6xLm5uMK0Cl2mLNkY0+f
dRCxchtqleUFFBbIk1zZPJ/BoqMGGKmLXSFTouU5wq4fSxj/rYGWr2U0WdmS9zJLUSHRwgc5yi8c
OGvNp94Ywa/ZffvJv+f1Zb1rUWF1D6dKjCU5uy06vLPQM3bZaq+qPqz/NCEOTmf1cpqI0jGbh5tb
XPVJOW11HzOBajxStvaHLnNPNKmQb2QNb7lGvCcPWVSLdSGYoQ3ExYmMHMGkuQFZmIap+L35jo9X
QDM1M1aL7jUceJc/Bytkb00UNg0WSCenKKUL28wuPPqL/nGWfOJN2B9af/xg94v1rCfY0FQzTEXb
BC3tR7IBI0IVqoFMtBUZFr1pKWKjLgKM2owqTASoJSCMXCVL52vmum4C8vKdXYO3eDKl/vY8kVzm
vNrQH5fZcbk5s8AFQHP41Snzilg08xEkzm0Qw4CtktVlRnGXHEciN1WEXCKRRCKLMqID25iceMGw
sEo7Vx44H9rPukGrC99dv1vvqKubPfbZcenOn6CudY0NeqAZl6FKeUrVgLmFaokEjDIbQqH63yZX
L98t9KgsV2TTeV+2UL90Dqbm5GTiMt1zeR+1rrpXUNJWtDAFE29VsyyvGOUlK9J3zQjaTGt/S+3Uy
ZRhmX3g7JiGdsekM0oqXghI83iJeDbF0VNPo0uRtO7GvHf54Z9uPunGZ/2UWK3+uV9cLfe/16+ty
xTTdjJTpjbN3VBB8Hit+xkJSH3SsWCb4yxijdpfyIYWNLiGnet4TooTMz1SMJrp6YX2meoBICydm
4da+2bWK1nSMbNgqfpfaMom1S8GmNRVhplAlwtY89pDRp0nKhPmleozuRT0Ztuc/P7o6SfRk9t+u
43wwqWFv7132nzKDVg/JZb3qi+p8Qc/k3MMhg51okqh7WtwSEdFgtHaREv9tAc8Z5wpWo9YFBCSi
dnMHqWGdcnAZOgyU1Y2DjQzQJCOLiRTWpQW4ZQ6V/xNNodrKaGYy3lqel+ZoiF7UhyNvm2eE+C2i
onrNHXkZjCWP3OTFPtYnrjDoK4Qm97DGe0gdaFK/B7/eZDbszgT53lu15WWJ6mTTTQxIIIXwSLW
CPMyema0kljWOIX2qmYt0VkBi2zjG9dkKAaTUFLtZyoMKFEamAsLCNKUvFBmFU9qYWZbJuKjCAz
uTosQLVpG9iYfNUvhdq+jK4ZKkZrL2rRam+bJpE+j1Pcty+eg10EmNja1NzucG5y3C6o71bt259W
fMJN8z6KbOYT7WdjoAhUgCFNfOWuQ6Foi6v5XdNhVEmTBauixqqUkHsVutKJ5jIcF3QzzWTxzVCb
JnAEf3STBg2zUotxs8t1kvJASBAPDxoIT9LolwHGscCOoJUhl0RmMq6tgKVvUtlacS8KQwu0MZs/
wRd8Xc3drZVUCZL3JSvXazyNLDC9nfK0//r5lNuwPqpsWo78+zSFB1kBQRPyPeZdrFw6LI2qbLTx
cq15pYlSgSsQyG00pi6zMVC856ULPW1uB4MBrHAniCjaJkAOSuKtYgoDftaZlZn1probxIJu7IIG
uDntXPzHPRXCyMyrNKmHCXJ+68Op+jJix2NBur/Vv0+yvEZlPI990JcOHP+0o2bT/+V8+ce+IXmg
27A+pJfT2UrgdFTN9IWCyRr7YOSS5VeLsWlnXDRIr/NOWAlrWenM7SwJgFusrClcWg0VIHWHBMu
XO3hrfWEY1x1qsUSW4zyoMMzLwqgUjRxFWSGV+V8qZ2rdFhgpURYz5OGZ2HBA+dpJjeOcK408Wo1
9PUDWv6i/4AwfZ5JG4be2fO06wP4mj5LnmbDVgbn/Wlv8QMzRgk4qCogL1WAVYkDEt5Fg0SeS2dh
M5a3Mpb5iCVJiBURYdDkfXKcqWKDbkOmKw1rZLT75R/SbVOsWA90KqokCSJHg6RgHhBabXEFEMw2
suCiEx+lwlbZqzL0Ppfy4B3FWOxf4AzwwFGG0Ntf4stIBqNMHj1m+D0d7C9dxGe620+XMC13Eu39
p/+ouaTbpj1pe+zZulMFfmFHSmMzfa4LGBvaVv80lzn0YIqg6X8rLKNlArb4eibJpGWJNjwLcU0r
dzcfFMVwGuKhtVHwoTMnSYcUMVywsooHFZkXbkff1foOI9M00ZkXlRGSdgyYwQAHh5yvWg6hZ/kO
L85RtjrCOPWmHMw1v2PYucVr0UlJvrOuQvaIHAynWA42sRyuWQ74PfLC/6V5tNuwPqSxSk17xM4f
LZlewF9z1D+Q9VfyoXaOvqNBQt7XVNOAiK6oLEk84+TU1UN/2/2XsTKNvO6jxwn3PuUPNc9WZNS
MgSqJkkJIEGJDBgY7Ad2+nEjtOdWuk67nYn3e603Z1e9ortFa/EdjdZnSwbD01jQIGYCWFLQkggR

gmBxBAQAiSB9MaaXs13PkP/37f//Z9zn9zdiWPj9+S6WqWqd+vWvWf497/3/va3v911Bud8qjOqq
cs54Q37ntvbhVZwv+e8qU6U6w8UNGJtFd00WZ/PZ6KDPSiABgG3qM7hU+hrzTs96WzvysZymCwdP
cCump57j92RURl5ze2y5Iy16z7jwLGLpFEbka2NdZmeX5C2M+yRbiYERJecPPnM044evXzfYPeN9
QL2rLXoq6QOJqnU4FmdgTSjPWmnkWxtrsg4BLMR7sLgOGpRSzMwMBp7X9XyAQShPxaG20XTuft3z
4WtPRAaXNg7gMvNYvdcQY+IEhHApYxT7fwMHkydqWpdgrD4Hj3sYK8tdfc+Lz16SBojTfz7rnPW
B9fkK03vElmD13CSXiHL7nIHx9dVtdXZGFhgZRF5FljYxOQRW2OTjT3veu+sV7gAFMyfgpKDGabs
ctGEqk3R6SRtqXtDG53b8sZbxLqsCAtpH7IF0iBGRUchHivDG/dP7pAceERC62PKpqbmJQFGQIGM
5t4ojqjXM7ZSTOKeKdQh8C0ObEkp84zLkxNyOzCPP8GzKYzeU3Sy2+Uq268ReojU9Lp7MrRyy5nW
H52bUlmIjNkDh2RWZfH7u5uClQxtjbb1/zLX/3f/sdf/Oe/8pv7d33fWC/Mx8jkRtFe6z1/2QS/L
46dAUWTUnciecztyV6rJVsg+oOBRJZRSjBoYKUbWHmU+NmpqlCIJ6nkgHzUWWVfB3FQnBvtdpyqH
nslw8LlvuASo2kA4xo7LnTe2ZLp0XFZuPiYy52bnLu67sLqztz3ycyVr5JLLn+p89B9absN4fClV
9Drb7h8dn5h0eXbPXKn4F13dnZJm6y7Y//iY48177rrw7/2R3901/YTX3ny9/Zv/L6xXnCPuDayh
dIIZFkgjo2eVlAlayMjtCeEyNvOWJETUofJhbxod4s5KIp6hSzn4JFSmaVgGxtkW2KX3DZYTI2LC
T4BIk4jwlFSOC+c9NCv2nGeu0bD39nYFsxfPHrxxTI6Me28ciotZ/DtuUslPnyVXhrRbTK7uOTC2
h0ypo5dcowh807etizOLblj7H048twMgKUWc+N6vSmf+8zn5F2/9/t4bbMeJ//nK1/xstNf+eqT9
+3f/X1jvaAexchoBlqiFBHE/yDlENXJ20X/60j0pDPcSNrbLdnb2pReq8tBxiA+JL45PXZ/h7INx
80AgIpEEeJcVR5qIEikXfclkJqz6LpvwYME6SBH+LpBPaeLjh2Wqel5ajrt9TLZmlYs8ZfdyHGR8
wePyMhYUzbPbsvEzIxMT09Ly3lOzMxZmD8gbWecSaMuUzPTsr21Ta+eOIP+xCcekve85z2y5Twvl
KayetF0h/qR66595bVffPwrX9lfAf+R2IYJYe0//vIeg6c/O5ae+fpzSVRbjDAlDmqBtRpb5nKIo
EGWFHlpyvM9hL+7W9Le3ZHO3q70Kb8CYKmnYBGKq84SGB4DHUZgDFUH56ohjoaRk7FXfRi0erJ8e
pn/PnrskDOyOYknpmSQNuRsx4Xgr7hVJo5dIWtnN2Tp4EEyqs6unZXDR464QxyRldv1lmimxidkc
2dbRkdHZWxiVNBx16glVXOG+sjnPio/+7u/K9vbuwzBGS1gqFVSx+vXFpbmrv7UJx9Z318F+8Z6Q
TzS579UH3z3se8kEh3lTFNoCkNUM5hPVxJ2xghb0NDbS1TYeckUjeH4GV0y3ZaCS/Cg6HcF4Z6Dm
1N6XOgvISRft830+oasn1qGfoRcdPFhmXFesjE+yybyVecR+wdfIQeuvVF2yW4qZMbln13nNdvdt
hw4cpi1142NLZkEc6lWl01nnBg21XTHvbJyhHEBV03jj31e7rzzTllb3eZYS21ScNGA24hqids86
omMjIw8euDQgTc89OCn9iVK9431AvCsJ7+eFM989r1I8qPssuHsDJ1KDM+ZNEYPqMa+VEqLghLY5
3gLtnhDY3inw17VnGiwB55AwnfGmbW7zhNvsjF888yKM/ZUDlx6iUzOOQMDIypusnyzVxXI4xW3y
dzSpXJ6ZdkZVV0mXUi7ub1Nr7iweIA9rlubmzI7t0Dj21g/K4sHllgGOnnihPOWTZe/tuWxzz8id
999t5w9u6klJeeV6y42j2MlzYH1hC945vHx0Y9c/tLLf+yDd969PwJ931jP70e+9mzU/8aDz8SSX
8Z5NxxgORT3gSBvIm6MSj0wwdCQiDCX83IW90B7u9dhEz1ILSjWDVD0r1B76bem78LSLsh19Vbrdv
ow545ucX5Jms85wViQ69KQpq5PfJ7NXvVZGJ6fKzKntMjE1ztB5e2NTxqammZ/u7e1Jp90S6dk56
XW6BJa0ntqXkyePy5g9ec1/7CY5+Tu+66S3Z3W/oz4pVb4VXpUZuS+HAYxopjmgZgYf/9bf/AHf
vqXf+nX8v0VsW+s528YfPqbUfqth56Ja43LEFZChBtEfRgdbg9GZ9Qm5t1zTZZBUCBFLloMupRvA
Wsp7+45I24R1AEpApPSe6096bu8lvNqID86MqaDkqnNFJE73Jq9QvpLr5DxQxcxTN5yOenU/Cx/f
3Z1jchvs9l0Ye8GDQt56e72Do97enaGBry+tiLlxoisLJ+Wxx57jB6111dWFWHJMRLyh2kjkPKvTId
ci+NGuav8bOYEcSF1ZPvOvRzz/xD/dXxD4afP7ehMPfVwy++SB4RaK9FRDxzRVUUrdeRYGIItqYW
+N+D201Yd8rJpP3qKTP3RciZvBmNp280ZT6OLxyze47GAae740f1M3J18rY0StlZmJK1jY3ZG93V
+YOHGBYvb01K4eOXcT3PHPMDD1rw+WkZ8+elXFn8KPjYzTg1t6OjI6NyYkTp5xHfUQ+9MEPSRvMq
rzghlPDdACpa7N7riEwIgSgWqQ8uvOFNhhQkYvb2Wv/glpvtv3P70Zx79n/dXxb5nPS8fj/9f/2Lkq
iMTx2ujo4uYjwpeLhhNQIMZEuLHODc1Hp1gwzLEk6Jc5Usxi7Xo7kjW2aQkS0EYfqqzaLqqB1Frw
LgzNvbuct/0otfJdv2ATM8vStIckfX1dY7yGBkd1larxfddnF8g6QLeEmR8vM+my1Xn0VjuNoz15
WUaPb50HH90Pu9y1Lvu+jBpjcaisgnpeA2MloOWC3PEXlvi6FYsX8fbAbIeWfmZn7xU5985Nf3V
8a+Zz3vHjtPf/XObPGVi7VGXafF2S+gdO+8HEZjAKKJ3WKHtHANrXIIzslUGmi4jLGMzk6nEzOC
arC9/PDrCJn7IO5K6S78HIZOONfOnCYRrO6vEivCbBn8+yGM54apVm6Lifd2dmSOWfQvV5HWrttW
VpaoiGe8EASqI4nT56WRx/9rHzorntcTqzgFoy0gfNlICEFWiPKSzg0hMFJULPyFKRN3fOo71J1g
iMlacz/8vbbbhr5xMOF/eX91bFvrOfN4+17/2D+2e+evPElLz0gS+OTUi9UhZBeC2R7yJH2B+xEz
aIuJ4sX0A6GFKifkUoPzJGRLuWFCQJ27kLNOMLIjZpkjXEpLn+T7GTOQzfHZG52lnRFkBRm5+eI1
gLVBV83adZlC22TjQFTUzPOYHcIIAHxhddPX1GRicnXmJadv/uyJcefOI+8pF7GaqrJx1XpkCE
qh5AmUosrEa7rhdHEADh8HD46JzhyGxGE+5n/e5aeDcV1bWf+n617764tdc95q/++/7e/uh3/7x
noegEubZx5099LF9TMrsnDwYuf1G1KghKedabKteC0Tf1Gr2oMn8HJOeLyoH04RyDbWGGQRT4iVi7
ROGQNRh3EsluvS1srPblcmZKwmOjsiuy01hrAhp8X3XedCDhw8RAFo+uSqNkURGGxOybtbFN8j4QX
xj26uqqqTIyNM28FovuFx56QO+58r3Q6HW4UFnKPJE0CWgzTUBaBAiqMN9IG+ziJw++hBwXkG1PXY
ah4D2wO/Lts8F8/+r1HZ2+6+fof/+xnHvsrX9bZV4r4S3ycfuz+1288/8wVhw4dkme+/YxkUC4s0
jBekfPLQFDxcw6VhrYyzC5JDvRGqMPmAYVGZMoRJRbUR0s2dZfKv71D6lfdLp1+JNOLi0SCAQrhM
Tc3J51W1+WcxaXDbGdbObMsUzOjLnccoUGCWgjia4x0Z2sbBAZZWVslIvypT39a7vjAe8iYCuqKb
BzISfC3kBYIMr7MY+IY0eyO3yH0rfnaK9r58DeZL9xod9HAefa9Hz55/ORXrr325XP7xrr/+Et7t
NdOvj/Ku1PI3x7/5mlpb60SiS1yP3qx0DCYnThwsQCdB10aLaiHRactmcsni4GfmgiXMyiFJfcl

PWr3yLx7FEKoc0vLvBvz5w6KePj4zQ2kBWQ087NzbhwdtcZ8SoNuCgSWTu7Sq+Lks1zzz2nYJE7j
LW1NZmYGJP773/AGer7WGdFux68YJIUzHeRqzJfzWkopQFiI8B5wTgVRBrLa9j47sdJm1bG5iSQn
NG/QwShrKz0Zf1edvrg177qv9k31v3H9/yx/syXf2J39dRLm+Nj8h0Xen57dVe2V1ck39smoaHgV
PKM3pXDjP3YC/S7srba2pRBe8d52g7RX04nX7pc5Mp3SO3Q1dL3t3Zyeoph6sb6JoEjhKswVBjm2
NgYObv4/cyM87Sdnvv3thxcWqSRfPfZ79C48Xt42omJCfnYA5+Ue++9V3NRT2yAt0zzwutAZdQhV
sMcoVe13BSbD9lWCHOLnN4WxkuvS6+c8gvn3Kw1CU6hjtsYBQDWaHZ6vX933bWv/OTNN91w1X7Ou
v/4nj3OHv/Or4zU4ua2Cxu/8dwZObE5kGXn+WYPHZWxxqS4mJm1HDAexhF7TgH+MM+DQj/Ap7xPI
bVofF6SS26QZOQFyerdOq+1C2FEaAXJN5KUTU5M0OoSWi4vzNAiUYvCaJBnjzyMjY/wdQmXmp+5vz
m5uUmH7fHxS7r3vPnngox9176ceVQn5LpTF/FhsKB4JpmoiwtnMgCSVoleWSeNcm9zgNeIaX6dk/
5hyp5BFReINTQUGDq1k92m39uP+1296/fUPj42N/sTHHvzk3r6x7j/+wh7bqyff4QzqkjHnmrdT
L67sStb/VxOnNqQiy5dl9rkgox0GpJORFLHOMccmktQF3RhcNyQCCWZqM/wOfp6iWSHrpW4PkEFC
ISmE5MTDKNPnTrFhY6wc3tzi4Zs5Zft7bMyPT3JEPX0aRf2Liww7EUI3Nrdk6kppR1CxiVyXx/8w
B/Jo4982r2+O6TfxOHLkXrSei3hd/HRABQYaZjIp1FRSvshH0ckEIt14SQ0dGjAEYBqJG6DUBNvp
fmZoJARu00BkjTdQbs5Ukve6jaMU6+78bp30+//Tx586JMvesR4Pwz+S3gM+vm/GpubbULyc7tfy
IbLPTGU/LmVLdlcPpyD3W0KnEHqgTNrfG8qWPwQTSO8NOa87yWvl8HR10t9bIJ9r1jUCFXhSU+dO
aldMULM0gMMcXZ6jsaML5RpAA7Boy4tLRCNXT5zhlrBMHCQHmCoe00f/P675Vof+STD5MR3y5DTW
2+o10QLn6jhUpANVMNBLaJKLMA90xjEqzoeT9GmrGudZfIegGoj9CYkYEAqGZwNr+4MKemo2+d
dWh8mUqt3n+ZSLFH5uZ2fnqdtvu/nWfc+6//hzfexsb77ZGcAl03MHZO3ZWHZ6BZFeMAXPbLrlf
WVNzheWnTcbYddLojouOlQqUvS31pB8/qUiM0fJbBpxxrLnjHvEeU+EmTA+1ENBD+y6kLflcr7Z2
XmlCrqQFmQGeM2d3V0Kes/Pz9IgAR7BuGfoq6v4eVyWT6/IH/7hHfLNbz/FPJozcUB8gIZxTvYX5
9RMfAgrGhYjdzUjo7YTyfwdHW6V5hVCf51UxIh8Z/dc5Of2uL8HpREZQIahW17cLXXvUSeqXHcRt
pJCMFbWeeYrnZd94I233/x1d54/fP9HP35q31j3H//5Fzxp/Ea9XmuiINMrEqK8oOLV0hWdnNZW
d+RJReyxlM7EjtDrYNaCJAUPa4LV7uw9ypJI0Vd8UDNDhJymoaxsrJCI4Ax7u4AeOrLwYNL9KDL7
ncgQ8Botz3nXHXCR4BODq7vsqelnZb02kWfxfJUrr/gY/JU9/+lhTOy6McA8+IEgs4yWjXKyKNP
EnYj9R4YajIOeE58Ry+s+wUqZfFcGd63rRg2QleFAwm1G1rdUWRMeQZ70PD9wqOaMBv4D0LfIrbF
OqWL0dsnnef0uyn6WvcLvKnt7z5ti+7XPct93zk/t5+GLz/+DM9Wq3d6wdp76UjzkhcYiatnW0XS
qpXGnEGu9PJZGVjj/klumXS1rZ209RnRF76NkmOXSV9gg3bzBvhQeENEX6CWQSACOWYra0d55Uj1
l8Quq6trcgCOmncogalcHJK89jNjQ165YmJKfasold1yWeQIU//OWvyhNPPMHRkz0wpdxSAbDEK
escvt6k4cOb29hImmukHUOs+yLMhdi4aIkGBt/r6zxZhN3YcBAWjzRGeU51NhvoaMq689ZqqL4xg
RPwPLML1gmRN3CMsZO51+ILYFlzdGyq2Ry51YX9z/zk3/zxX/37f+9vR/vGuv/4T364tfVv00Hu1
tKYtNodGZmaYY42MT7KULDLnNap7V05u7Ejuy7P7OzuSb/XknTMec6RaXokGBm0jndr4BnhCZGTj
k9PMW/dPLtF4GHqaoIGCAQYlEJ4zc2dXdZcsTkA7QXgg1wW6g4jrlnOMRw+efKkPPTgw7Lb2mHoC
08Jw+R8V+gIg/jQTyu0wbJ+2qyPa07pJA/vjQ0Ez+M7+3ElpwHTECNffwF8FX6cJIEkoMEQC4sUg
KJnpZGqQbt0i5XhlEnHeX3k6FDN2Ntp8fhxbiunl4+6cP7nz57deOq/+lv/5YtidMd+GPw9ejhDm
3X52EVxTdX/evCKS8dkY01D0NWlswwbztuFrG62ZNYZ5Eg8K/Vjb5B89pjUcq1LIicFiITFi+Zue
FiIaLdaAI72XNh7gF4QCxYGC8+z47w1qiYLBxZoJJs2867TjCEPnn8BI0ZIEjx48fpJT/x8U/K6
eWTVJqgdCkogL2UzegAebSh3CO+CG2LIhga8LMzxN6d0OM1TwxCIBnflbnnUkxd08GD8/nnbcFK
QLeN9HmVw6V5vt6ehPeg7XcHMOehZKO7byrITQ6izASBALj3V6z1epcudGoffAHf+BNX3fx7W988
N9/+Jl9Y91//H8+nJG82621RYAo8AJYChlz3BlBj54QISEU81tuQU5dc4Mc/rGf1XxsQVrOm+xs7
Epe22PISIQXNVIXziJZq1E8e4eTyJeWdnKe6rrLQYH2YvHC02AzALC05zwrhk3NzE6RbHD65CkSJ
WC0CKkPHDggjz7yRfnCFz9PxQkYLowNIzjQWctWni/Fgu4a/F7pkIn2xIuCSon3tjBEM1Z4cfOGY
E/r9YhGCNULMqZm51TNEaqNMDqUc2D1+I53doYeE4CKPWNKc2J6e0xwr24fa/1WCnXHaa7oZig1J
U13/V/T6/Wf+Gs/+vYvj4yN/OD73/fv2xfaGtrvZ/0ePJwXiLrtzjNuYV5WbzZkc23Zudo9WX/6a
7J58jvyhSe+Iu/+08w+lh+8//+gBy65OXsmhmfGGUeeerESRLwL7vsMi7Gne1NiqhRA21ji+Ewv
Ci6YNDOBoNGSWRvtylZ81PBuKFICokVdNLAQBvO+M+sLNOIDxw6KCefOy2/8Zu/5cLwVRoMDBvSM
FkWsem9YaMkBz0NaaMich5NsdDAJZaoMKmutyedbp+bCZ5vNMZcyO2Mtalln5OnT8jm1LSM1sYki
1OPCoQ6DnHsXptAlZ+rVqPLL1tEjMPU2ZzSeIZXxQ/z7XZPvabiPfq4+Oja24zu2NmavJ/eNfvv
yfbz1n3H+Hx8Yfuv8Yt7AUS7QuImHXcgnNuYXxaDLxyhVx11VX0rocvvkyuee2tsuRC2aRWyDNPP
0WE9n+hFF8tFF13CnlLkqPBsyF/Xz26yxAK+LlBdLGWEyAyPnaEsLs2TWLC8fFrGJydkdEQJ+sBTg
Rgvr6zRUJFTLp9ak/ff+QHZ2j7LXJHUP6ghZpEaZaI8XXwGPCO9X80YRzWVoUfDNY+Yg+J3bIJPc
xIeyMBKmlpjbejx49ymJ6eY56aRlonx6GUKLOW5NrHXkiKQMPBZFI1LVRQuylRxAog3w3Lk1AMLY
8SRHpcUcUCWke/2utni+vrGzz13/NS3f+on//rf3jfW/Qcf17321RdPjE/f67b5qXTQ4dyXbn8gK
+tbko3NSDRzgOMpXnLpMZleXGAoCBkV5I9o/D5y5Jh6OZczguAAR7m1u+eMapekAXg2eMdRlHlqM
T0tCRAzU9La6xDxNSVCGAdlR52HhUfrd9o0NgBRj3/pCXnqyW9wgXPcBvi7VHTI6TFhtPSabtGPj
SiVed9RzoGHNqXCBkPZgp4bXzAQgE0oL2EjwbHhuMBBxr/BH+bchQOWRD0gzhu/w6PmWRxKOchVA
aQVXoiNGwg9robbMFT1K2tIjGYCC51htAid0fWDY9ja3r7s+edPvOutb7rtmy48vm4/Z/Or+nj5y
668zq2y33fGdfHVV189hUX0za88KU888T157JGHnYdbI8jz9//bfyYvf+Nfly9++Wuysr0nuy6XB

KHBwstedyATk2NchNtbu2yHQ354YGlBife7W84w51UIdKvFGilnozoPCmOjav5um4v10OHDnFh+4
sRx9Z6Sq+DZ+obc8+G7XKib8z0R/sIgVEDJvVXsc1CEkT1sHC607LtjQ9qamygavGwjYn6q3m4QS
i8GVCGHxkYAnWEjRgB8CqCRKIkCfwtDpTcstNUu9SAWjp2G6gzY6s04P0Tf6JXVLh4XHntv30t7B
LugTwWes4XEeH2322261OFKtW+dOstNz09MTH6w/fe9+CpfWN9kT+uv/66qNVq/fQg7f4vSR4dy
qNi6hXXvYb54/bmWXnyqa/Id595ynnJnCUV1A0/ds8HZfQnJ+VHfvpN5SMfeo+Mj40x1MXk8alpp
RH2vVIgctN6I+HEcXhePDczOyM7uxuS9TI5fPgoPQnC3unpeU4wh3FMTc7I/MKs7060XD66xvwWd
ERwGLGu3/ve93IBg+TP1jQsfBiNW/C5W9EwDkVwm9rqBoEzF45iQJZpJ+Hv67VmmHsDY4Pi4Qja5
pWHBqd4fXWLs3MOHlkhJw7oitgYrKZKGWryjTnBhvv9INOG+4Z73/5AGwfoaXMJ+k5leKzDtjBRL
yb9MSploIBKw7G6a8OmCE5wd965X4QcN897UwpC9b7xuhuv//Lk5NgPPPCxhzv7YfCL7Pha61+dv
OK/ePk/2d7aeqrXbb+ryIorB3k2NeHywptedxNf86UvPsJ6YL05RrAFdUcSG7od+eKnPy7z114mf
+tnf4GeDvnk5PQEEVrIfgJsYq5Xj0lwGPeDl0NIjA4ZoLFLhw678LpDLwLDxeLFWNa+VGgAw9PCK
FBLxd/0Wh2GxB+++yP8HHg3bAo5SzNF6J6Bp7Pm8Z4L42G4+JmIbF3RXmws8Ias7NAGChpww/8dQ
n+E9r1BVw4tLfK5OG7QMxK1BYoMRJegmbalppjk4/YL5rP5MAaE12rDu3rUqjaxeLTLAcadIwL1XX
+VkfHiiN+X5Mkx2mwAGfoGbDfp1OshdupLfurm+9fyNN7z2nd//ptuTfWN9MRjpdA+qvfpV1/z63
k7rObfgfz1Nele6S9rEbr2weIIE+KuvermcOnmcngWe0eqF+BnGMT4yzoWzVHJaZhcPshkci6nb1
tBwbm6BwtpQuQfX9+jRixja4r3PnD5JTz7M001pFuSI0EfCdyC+kGopnNfC64iCTozK2soqFzgAp
49990PyrW9+U+feIPSttKxlvkxCpQeXC2fIPV2+iVwSr4dhaS4IOdGMnF0VAe8wNEXtF0a/vLpG0
gb05eDBgzW0lSltLXkqFSWaoU5roTeeG/jGeiLAHYQ+2KzIg3EmvuZrx6TnoSqKrA97nnIHQ6bdh
qahtAvj3ebRT/Pw+4HPeUHaqDVGfGdp/nOddu/bt958y3+3HwZfqKDRta9s9NPB77nd+za3cBedF
2ticdTjOnmtE5PTXDSTziBe8pIr5PFHHuFCSpo5yxizc0uytb1B4AUKho0m+kk3GEaeWT7lQuFJm
Z6dInl/d29Tei5cm3He1lBgvAfKNBhngcW/urYpkxMT9DZnnTFOOWOGR93ZRtmkJUsLSwRVEBJjc
BTEz06/70H5+MMP0dsh30V9116OnF9npG4TgHEiV4VxW6gLgR+pkcDcQGAldNYbDBCpDjhCwY6s
B/GC9YRar6IAuAlNS9VJUMz1BrE1Dz5nyExOMO+NEWP635PUAkhcb0RCBg1L8LGzNjXZRFRaK7rj
i9G7NsgCwpEicwbPY4ZXnu0Oeo2nxDZLQiEgHPE3wNB0HwGnecl7mvf/3KV77yZ5uN2j987AuPf
2rfWC8MIx1znuM0t+Be7W74UajOU6SMO3wszdFxF3JO09sg57v66qvp5RCKAQvtb+3IqFvMzYUFA
bV3GQq3uju5J2huuDmlprLQWdJKgDLcB5k0RkkHiA+bIFL7N4HKOqxI4dlY2tTnn/+eYJKMKrTp
0/L/OIcvTYMs+Y8E8TO9jpd6bT3fGP5lnzta0/KZx/5nA99u14mVMNekPNRmxxplkO5w8CuQEYwj
ahEAtEeHhUbFkLtVqdNfWCoVEBgTUs9RUB0VSQtCuErqytpQV6wAkU1ft1mzLoUwGbPqjfwGJDi4
chHE5h+g94djCs8n6e+Yd+F2ohiwEnGBjKG2beYe9toaqsduojabR9iRrohFDiuHifwRb6byG0MT
WewV/Z6tQeuuuqqr4+ONv/Gl770lWf2jfx8DHen3c26x3mPq91iWgRwgvAqL7TdCyHg5NSM9wZpY
NdcscklmjvBmBHSuecmnTGuLq/J/MwBWxeGgxallvot1x1RjJi14WzA5ZhYIDI/1yFfrwWapKd7
q4ziI5bXNpzSrTXedntrRYNM3eL8/Sp59kWh+MEPREGDK+7vrYhT3/r2/LFLzwhqytnAnAVRZr70
VzR0VOvc3EjnGQ3jDOAtO+MlueohqbC0DVHFDKSUB2BsaKeiw2jwd7VUR47Dd8LjTP8lSgIqsHLK
nCkWk9gSoHNZVkaCkg1jBUpsiC0hs2AIbH7L8sHAT0eDFQWJsbwaMi0RsIIBueD59Dq107b0i8oj
kLpB+eoIXeJXBdpLxxfonXmpvv817jvT7zsZvd9udlov01LX/5qa99YzwsjfeXBbDC4qzfIL3cLd
xEghZUaIGqG/1IQcmBcCA1NksQW86jLR3cwb8YZEAXmcnRSlvNYpmYmpb2HFjZxhgpAadaFum5Rj
U5IMxrhOAqUW+A5Wc8cneD7biGs7fTkWfZlXOOMaDEKxsW56EDB11/3d3ZkHn3czYoZNV9fODAI
hc/yPlYmMdPnJLHvvCoz1MVYTXvqKiv1kvJGqpZCFnQ+yLpQTtdoPgPA4ABYSFDGhUGuDDjzmWgv
wNbiSE0qJR4v6QMf7Ex1D0x33ScOHZAe3MYTcOdO/JHk4/p9z2TCSJrmDeLz+cs2j4nu2e5gUxK/
K81G+zk6fYyz7pKSJ2088WxF54ppeCaoltRkemc20odN6ewm/jn4il33Lf2+v13r3n51R91G+bf/
eznHv0L10rdB5j+1MeNN1x75FWvfPkjnU73W+1u90bnQRbZp5nUedMt15pxng5CYyzss6aXhoWak
YwwtCkyjCakhgUEXk+C7pYlabvF0U8zKutj7mmnDxKAKth397pEKMFOqr8n81v8e2FxVhYXDsiuM
loQ//daPRoXvObZzXVpd3Ziym9t73BD0YReVhjo8eMcjQHQ6e577uHvSi9XoqjwonqOg6GwU1vUa
n4CQEFxcTCUzIgKJelSoXBkbMwboII2NPxCecX6OZEXW4sJVunGFsm5tFfyj0WfQx5LIw8ibYSUV
McJOSbwdXralO122EDxWnhPSK0SSPIgmtZ/1UNj84VRmrZxof5en6dl5oHPjHPM85jUS5gNXtPvd
Ra7vcFPn93Y+tz11776F37h5/9ptG+s36PHzTfdCNTlpY/s7e19w91UZ6TPVM+5POzu2HEpJzJQ/
aBZMI/qTd5sHRmRBQKAtYAhb8TcUws34X2npyakM9DdGojP2sopGR2fYo6K/BahL4EdD3LASFHki
ZnDpRSM7fL5ejRwxw3Uc6BkU3ifc5uEaianZuUDfzcYjRTc7POUHf1d377d+Xs+oqWZ7go+9wcc
G4Q56ZhpMW+aGwgXczqLUEVhFHSYL3BF0JDW7eakB8C1DLPZSUVGBxz3khC7jPgi50PwYMe08VA
y98J0/d85HtmODwYm/Xqkih3hHpBNhU2BARTaDrBqAarp95TzSY0kEeQnk2Gehu5I1Wv/IwIcwmG
0v88eQ8VzEBcn7HZ/S7vct299q//NGHPv7Vf/Azf+/AvrH+RRrpLTceueH66x7Z3W092W53b3Q3b
Mp2YeYwLo/DLk3CgLuRE2OTRD0NbFFurBfepvqgLiDUNudmp0n3Ix3PhWWTmG3a63CiOIz3z0116
vbWaiPOo2JiCsTPAQkffOGxUQUWU4GFJTOA80xGS4QtPIQSQA08Ow4SXHZ9oMi8GeWFiZlqWT56SD
991t3zr6W+ShKDKginlQ3HslAuNmlyY2CQqSq00CUPSnDbyPzEajSAYc9bXVjjoJ7kNaqExNqhRt
wlxWHJcDwaHDYyIsq/T5p7KiG4ZC71x3WCJ7IzLJJRrzLgL77cgy2qGis9V1YmYhli4kDf14zuMf
ph7Q0tzzXM1Pxf9wu9IRcw4nR0RgZWOCp8xZ/ZTXhq6bcwcWM2WvH7Tba7XfP6xL37qR3/kh+r7x
vrn/HjTm26au/6GV3+y0+59o91u39jt9aaiXhDNC5fYxpV6QjrZMANlPW3Eg3gPpZ6gCiSbHvXbz

jgQcqI8Mj417Xb5lOAQyjdTKyq2vbvbKZPHv+u86A7J9zAetLNNT03TU8BIC7/QsKCAupIr3FciO
9T8dzd3ZbfTdZ72KH+3urou086bwkhOfPc5Of7s8/Lxjz8kaa8LzJMLGYvdBbEyMqKbGIJAykaix
0sk5N4KwJSelnlu7jtuEvGEib6cXj7Dv0epxsJcy+GDARSKNsMYYaD9IvMdPJGXgamzZKVSL/UgE
2N/n/hSkG2/wvVC+AtkHohwZ9D3gFPmQaw0vD7z4T3vaX+gqheewcTni9i/zgAwbySo71ZYU/Z+d
kx6a7Sp3t2fK5999ruPvv2HfvDPPST+Kwkwve1tb5zY2t79k263d7XbcRcHXW+YUDHIFeq3Dg+UN
ah2HyW+j1PvIPK/pYXFQBKA5IktTFxWeOTjz52gMU+MTMj2zqY06yADoCulybawlvMrMj7akKy95
wwJqhCdIPNJSU+fM3H3d8+326jTdlzYnMmmylF7/QYpgthAjrhcFoa8tdciS6jd7dHLwaO8+73vZ
akIQmMpAJaBTnBrNrQ8k/v8FTVMhuv+PE2Yu1Ef4aYjftobyzn1ctod/g2Ko0md4m9YU/XGxfesq
eEhIkHeH3uWEcj1wSsacR9c4KbWUw20s5+tm6es9apxsfdVLGTxkZL9tOPHCxhFSHQ6FfozBABwK
xMxbnPO8hvBNEQL9RqPk3/mN2yytDwgp1Pw1IHCgyNaxpAwJBguc3rNiRMnPUh+9RP7nvXP+Piht
7+lefMtN9yztrH5nVarfatbO4sD5pMqIKOymikRRtwwm9od+bCPYy0KDQfBKlrfWAti1zaYKZIyB
4PhPfvss86YFDhCt8vExLQUUDFwoTXyz5dcdinDzrHRCff6jgqSCWlvLmRVhQWaj5BYQVmk32tL2
4XN4AOvrq/JkSOHKCW67gwTxnpgfk62NlscchozyfV+8P2y7PjiLlhuvxUAMOon+XCUBP2aL6EYl
VDri964BvS0ZdjPlRrC1DMrp/m7pcWDNHCEpamvprR6a5RBu07MVQuPyuZR8N5kJKV5aFq3nD14F
9/dY5taj55UPZt25GSLAWV6PzTSUc+Idj0Ca4XKp0a+lFRozYgGaU30FuoWvtQU27EQbk5CvozXI
VrBd/weoFa703n7DTe89tZ9Y/1PfPzYj78jeuPtt/yq8zTPuAX/jl57sKiIbep35cJ5lEI9TlqEB
ZJ6owPDhYbIEHWUuzDKENoClmcnh93w8uYXnNWC7hqEhFBWmOpGzV4MJC4AxrTloiNHSR1MatBUm
vZSJhERzJ4zZoAlvb0u024gH9rrd9z3lnqcQpFLTB3HMaOeCHJF7nKvpvPYD9z/Mao+4BxQE2UoL
3EYWWEe3IzEIgd8twc9dyIeOS7zVaK6zhW1l1f4PvPzi6yvmifS6mgyFMJaKG2tcPCQaEzgeEiQ7
ws1qFoyEkombi5ppCGhE8kYVBnr2AqS2XWv/h3+n8QqW5pHeZkvR3qvjJXBfxO8Ug+rYS7/WiOmu
BQv52aDzcAjyKzBFBuWmxnb5jaK5u5u6w/e8fa3JfvG+h/5ePObb3mHW8BPTfe2fz4bpEdTD5DQc
/X7oYaW5joxXBIJoWfprEIAvOBmwWDHnNdCyMecSfpszjbPgZtnYMRXv/YVbgiQBsUNhrfD541Oj
TkvuefCYJe7jo+yYwRGjxwSTBp4SRg2c1lsCnudbedRW/R0YCPBMI3WuLgwwxIOAa25Of786U99V
u784Ad9HVUFzvKiVHag0dvjT/XLA53QckIsRII8US0gp2YsXNTuc5fPrjA6ABlDNzEJmwDKWZzJ5
mkWcsPQOQMHhdJNrrIwocRDr5cGQMtyQzM+3De90F6fGBKnud8QmH0irSykRESxnodsZ/F+enliU
77g+3v0t4iRGsTBY2JXLTYyJftn18FQ7vK7b4WHHJXl1fLD04LLTp0//2r6x/v88fviHv//i229/3
eOtdveDu629K10o2RwgR4p0MaYeJSRimKJ2GAUBME4u8yRzM0CrK1LusqnN1+FGoQbpQyYbZCQeS
Tx98gznnS4uzQpUDcemJqXn3h/Nz6PjE5RrabqFLYV2nkTaY0nmERq0EQZv722w0RtoKwnyhd46G
AlAIYTLMOD5OzerOoMhnfATn/gEARdsSGiVQyJXROjrPR01TjD8KS09KuuduXbZYGFic9KmbW9oo
100CBdxbOD+QvepXhsNwmcgVjCFYJnDh76e7ECASHI/X1Y1hW38hoW6KEHBWxrDKXhmv3GUaKw3F
KK2qagckzfyJAobLX7mJpXwqNyp4J5mlFmLSpXryMxz9+EzJ7HHXi5VktbVn4q0nhz7GjJrx7ECY
5EpiwFS2eaCddTu9H7q9tture8b6//L461vfcm7Nzd2/003M0B/IqYSk/0S86ZnbJnirYbKex7wz
JWIWITldPA4Q2yk5sgQR5aTWiohr+2yJRBSSezx1FPf0vcstIsFYBa8339wHndhbp7h49jYVGiqx
s2Gh4RH3dxYZVmm3+2ozOeoAj1jE+PSCBsGhkWR0bS4JIShjrBXdw11WU6fWpbnn/s082qEv3CTj
WZT64U+pEQt15rKufBE827zrJrNDXW04YkKDFVMnCc3uS5H3WdSdnRUwR5sBCTqlyJP0UvO2fRKe
ZgAxnlvzzwalMJM29qYP3rjNxD1V5xXAibtUfVPGOpzRR74bTYN71H/ndJomWm2G/KFh7zc2L1y
omyLsLGDSHYw+Hcb5ISyjisDRdq9FIMwsYeFeX5unM9un5287f3jfWcxzve8ea522+7+fHdndbP9
fuDqQEL98NIItk5keoCAenFDbdd30ALeolIaXXwZOPU9Y2G4HrcXNUDKqeIRVE1PdGbC9QYRrOwt
OinWuuOe+2WHDx01JTE8YkpGg28N4wRpRfUVKcnJonEKng94QwHdcseRbknnVfWcYoJPTTa8PCZ8
Ngfvucuabtc18LZWqU8gU2o6ZFU5m/GZPJyKn1RsuZC7dg9Z94XXh86w/OzC9Lw18Vea8ZS/bdt
ghfDWiCoQft1WMOF7b1nXyP3I/iQJRiaoUEfoTemSLfRalbJHsaFnVcCxeUhtliCGFXNUUGOE9T
eTNFCjsb/lheRRCaNxeOy97jbK7LayOyVlmJiRZSIXchv1m513/s3PXC65084Y3XBu5hf7G717rX
4y5FX3w4NFx1MHQx00uymSvny/Ck+YqwgPoKuQyU2XXGEqYAgzwtDcarZHI3SUFz7Q22mTeVX3YD
Q/cWb8w7d8hZ4098MBwOJbPf/7zcsPltyhBwBlIa2tb5haW5PChi9lvunjwIMkKlAt1HjVNMepwx
hnkeFCIGicEk2lvT/NehooM8xo0cDSSo/H83e9+t2xsraeApl6zJu2BIrWNEtG1sD7wc/3iqoJMM
ZBiX+vFA80DY86jo+WNC4i/U+DGWFow6CqQ01JjHnNazJHhVRDUkRbjnepn+PUC1xJBaboYJhdHM4
JH7RjbdPfYzghB2xlKqQlRGTabD95usecQ4qtIbY268WtaBV4zoiQeZhsOHX48Sj2RXG91tMyvoT
Xn9sHb8ZiMeVU7TfHFza+dn3BO//VfCs95yy7XRdde97FdX188+s7F+9k+mZuZuXFg4eM2g27vM5
U6XuQXrvgaLOXRKcW1RsBPNReR3t74ycoqcZRTKqFwYXtNWB2JBr4O80hqZq4Vw+1790XgSqZcLR
iJf21A+7beffsrlq3Ua5uTsJAtRn5NrXvUqOfaSl8js4gEZmVatYISmOuhYc9FxaC85I0WbmQ4kT
nQSm3tdolnj+RjwBdt24Ycflme/83QAZsBKsgWK8wHryYynGppWG75Lj5H44y+BNTAAVYxtJoSui
UfKzTA05PfoayZDZZYh2qAvDw0qnTiqwB8PbYwG9vHnQtLEWk7xsb1HeAvvictatwFCBYxiT8O4
y5rPlyGwxYC69+Uz4HdZEwoOcdTa5Tl2U3c5HK/2cUWETxd+vzHfyXC4JtuevWBvb32V3dbez+/t
7N92cGDh5vsBW21pNUBQygjyJJVyi6W3zD0Ah/U1+5YiPehDksZqQ/LaqrgrNw0y0p9n+pNOXeBm
7Eqpc2rAdquberw71499vnH+drJ8RnOtmkNonL55dfI0sGjMuXyVoS6oBaynzTvs5QD8EcR1IR91
ZwG3mwSDTbhMmNM4XWHXUj9hS98gV4N64MC3D40rPm5MkbXO5dVd07GY6WSEA66xQdDxWehkaAa7
prhV91FcSKha8UMylhL9LKIQvLII7P1oU2R30MiDqFvIP/HWiLBRLijf/X1VPHIciVNsfsfwTuoG

hrUbPKpoSyEex/Fobab+/CXDQMobfFLN29YZjXk12vk3i8ajsKAUTBw9wCku3YHX/2qV4y/qI315
puvu7LT6T2+1969xnnQ5uzsIj0Gam0wQiXNZ6oMUGnBANUP+RkU731DRAn2Vkc0pbxUyhtMRbxaI
yzoquxIqK/5f5eopC9nRIoa8neF5TFCbw05UOSaaAyHxRw7dgnRYRgchMzwPgshDjKcjT2pAiTW2
Tz6ngh98TqEnlicVkbC+Z4+fVLe9a53yZkzJyphZhyYOQh/oUNs4maWlxpvdYhCqBiGlVuwAYBiC
ama0Enj/7bba5cbWVHmqQGB+VxVUD6Svmikf3J+ayXpwfJYvq4WaYSAnDHKK0OY3QaTFENGHELVI
g5lFNtw9PniBfcxqpDzbRNHiYvhea7IPDxuNuh7MoW+T02UTFH12ASgojSUhhTwGgTPip/deUHb6
R+/aI31DW947Vi/072v0+0eRc0RHmjMt2CpsWiIZ2wZgkV+50a+2ScXNOeF7g7SACaR9ZJpfyJRR
W98I80xT08rL00VSLBFXQp0ZaGQH3k0WXD7RZdjMXJFIffffz/pa+AIH3E5Kjtp304+MzvPRT06P
sJIYengIRotPFnarT6HVqFkg9fjmAB41V4v59889tgX/fuXIWWK7OKM1Ch6JCVSK/VLy0crXoJM
1Ao/rOVzhk/dImnZ+aoRhiMJo+GjJ1lFQ/4WL4cV/SALQJgrhlFQWXQrqkZHTpj8D52jxURLoE80
AsZLklczVd9dGAePOAGIkP14RJTkiK4r4U3oCDMHispBrWekKWjLQJGzCbHzIdzsWWOd8aqABUJ
WzWTLiCdfB4mm6d/vSL11jb7e5D7V73MjRoY5edcQs41PaI6/IX3mA8n/rcSNkkuacLeug/KvOnz
LeokYVipOxCmT3GiTg3LzGjDQuIjJo4UOOGmD0596xxEVDiD33oQ0Rz5+YOyDLRywgIoYEceSRKM
Kr4V2P4C4/abKixkG0EL1tvBpldG/iEQ9nc3JI73n8nmU4kaGSDwKRJ6koKYD5X03CVYWjqsFkKhC
bsYyvfmMOG9QSVEE9vU1AyNIYS8tSgYAY0+10jHNrIkyH9KyDHJAa6UbWisaZkz270N6G8R+RqtD
lmjmr8Hw/fBVAyN5VQdr6GRjhdEGwyG2uMKgoxJyKktH7W6tIKPXvtYU19z1gHRVuYjeURsYynz/
SLKFU22ETp53GP2huuvS150xnrj6659h7s4ryZJvtMhKUASC5Mk3ExV2tPWLdABsfPZQ1QpEUUMG
4mqAVRBAWru5JrzIKcyRktZa0yGSjPVckDZeeHzSo8ActQv8houSFON71Pl71d+5VcIDB06ctgZ6
TiNEIR1GLGGjVFQo7ednaBSXVvjSfaIS45ut9OXz372M3Li5HfDqAnzZNqipoaKiW2sH8ZKzTojq
VL4qnNk2FPqFhpUCXudLumSFDIb5F4su0wJlEeNDWA4byy5xTW9vt54w2CpuM5UBeg0w9MCQqsa+
4tgsDZZXSJvDDh23t08EspK8PTmqav3zDZYgl+e5C+e76sEmEHoV418uS3GZsExARZF9MsSkEjo3
02szp5Hw17buMZ87SCwqdzvpnr99OCLzljzNpTXLi9tIgQhEWB8mjVT8XqySmzIeKfW/atc0tyK+
8ZOqXvZjiINO2Z17AJ6STmuIfKUM18PjeMyjKveeJu3wsVRKPJHipkoAj1cLtB6LX7/wAMPyvjv/z
b/WDp3Egxc1bVFDvgRPZvleaL524TwaAOh5vSqDhcArJ1fkW3ffHQgag8q8VOS9rLHWtauF51SpK
cNY7N9WI7QFz03OWRaID1BQbLqNpVoSqfJ9EdZmeZ/lG6gJVj2X1jS98cb10FJoZZ66Jz7wc3H8f
U1NkjBIWUKRMPKSb12plyYxja2MaDxqDEaae0+8X5nW6HLPUL7lCUTz1zyYFVcK3tmC2WiRawL1
/xaioY6fyzKkCIL3t4UQ6IiHyZQ6PFpua9ffFEZ6+tvuv4fdfv9S7hgKS2pXsUYJODaYjfUQnstl
CqsJxIXvyorQvqgD3VpxOg/jSTA/VSPbzbCuIhq3mPT0bRNzgriFSPOxIfDJew/tOnkeuNTP0jppq
ae+Eco/qKSOWF8dpabt1PQsm85R5mm5HB3K+SgtjbtwufD83ciHZnjfo+/6I8xroReCprBlmIALZ
KUR25iqi0vZOdrmZmE/v9cVmMPfPv/8d2V8dIw5snoVj5pXWEkcUuXC9ixVgAXpBzYaPEcv6gM+C
9v1uyK4WablJFVUkCqRGrkpRBCW14tkdVNff2WgnWp5qqRYheMeCIfCntJiSKTAGpxaLpRSOt+A
/KblKH3Csp5+icKsr0njnxEZp1YVeeQ5fwnPY/85sD7PwhKIu71t7xojPX733TrvFtc/6u7WE0k+
kAiVRg6DjxTekWfs+V+EreGm6myWxBKJX6SmRQVKF/Y4ch6q/cqlPIo0G+pYxn0huWeYiZDobCGm
ekQqBG8rUmc2Feo8ensUmteP3bsIuZ/MDiUn6zli6MJ3euBrsIg0dsKsj5KNcbRRR4b+c/79reel
Y9+9F7NsdxrEWDY+Duk+OANq5onktqH0k8+GOqEwcaIPH55ZYULH+EvJ8D53NvCRer4oq0wIM5aK
lIpz2Ko/GOBxOHZOpSRsLnWG7HKp6CsU6vUe0OombG8GVIO7/0YHRX6d2QKZRKohmWdOA/AISmBN
SunqcGixs65rsxBtaHeelUpI+spigYiJb6mHPJxH2X085KqKP6e21fk3yuKCr+urCFCx18Uxvobv
/GrkfMwD7uTokocuHe5Aq+VVLBYvSl0dvqDsteRuYbvWQy9pdT7iaGP4zsmIhq1TWIwPR8suphzR
htB+b3KeDgJNQPVxvJiyIBNnQ830Yw2Ng9duXF462PHjnlpzMQrMyjfGHNjIO9CKdNYwzOT0IyU5
+Y/q+AG9r+/8//QNjogtrlyVGEI5OY6T4m/pVHU60PsHTMipBHwyPCKJUNBE/R3t2UODMLGSCA/I
BQMORo8nNSHcnfdEOPkPIWmcYmC1zavFXmONFQH+beehQQJUD5LjyBz0xWfHxsQZVFFHxCIMcMh
ZFY4RvE+yWDYjAhbYfgr0W0hWF9tjdNS9aVNKYoiG7b0LJDhGX/6sgHicGaKU+SLOBPAJURaATN
256/Y21C95YH3zoE7/nrsM1DBkHKjoGAxsZGfd9g9EQ8lptTA4XPPck7agWclJQ7qrME441dB6Wp
HEsHE9AYF5Wq/khWXHIT6tlyCrSaArYm4BqCHau5zXk9IrLr+TnowlLF3NCIADh3e2tn0nTUKew
AkGiYJK/j0RCXz0/o/JU08+SUK9Cpk5L9oYpfeG19McOKUnDuSHClNjNxcNa6PY56CivGSQH8anZ
2TC13Uth7V81cLg3HvyLLCAsNBzJV8YH9ir7AfAjvIo+VctWu9fRMV8eE/SFIuUobFFS+UGUQKL1
ssaVe69dT5ZylvkaY0hVKXaWhRKfrqBWC00DrV3y3PDRmwsKkiwSjHkye3cxc+rDU0RkYTKAJvUo
/CemMI+fUEb6y/90j+7ttPq/BSJDiiVVICBkHt16iGyQPSZMZyQcFfNNXfxfoRh2YjPnzR3UGTud
nTzpfAqTsCRPNisZkpVQ2hsFkUmrswzPVyIKETL4Zg0KPSZMZYQcFfNNXfxfoRh2YjPnzR3UGTud
j9MCXmsDhIujxOps+ub8r73vS9oQIWM7rTngaUkPffVUEoqSPbAq+xbfZT1F/c5Z5ZXCWhNT0wOM
ZiMimkbox3XwCvLj7w3fSViiaJNMDzb2HQwc0115Cac5aG0UxQlh5geKNOYJGEfmxZOJDedJFEmU
+jaqTCdJC+nxPlrEyflYXSW965cXCWdUfPpvCgq2sV63ULdG0djQhOahm6ec0klADRDGiBZAJ0Qu
bjfo2B+5QVrrL/zO++sP/rZz93pLmkToR3VAUXVBLW3skY6mAqY1fyks4i6u7bLhZqh606d5VqY7
gHc8WEySzxuUVsOFesyTppjMajUHqvyosqKiYdYS4UPxXKvOGsASDUM5uL02rRoVofOMNhxCFPxW
pAaAPQATJqZmaIXbeJ3knKEBTedrFwIDz30kKyuLTMEhndE2afwYyRU+GwYtQ2GW+Hk1i1PjeoKN
L1/L6+tslY7PzcnI/VGMB6r7Ya5qHkZ1TDUKZQ8jTwImGeVOTUeoSUqy3psXZlmHhwsqY4KMgU0n
fdQSQqpR9pt81EgL61QISP/uRIYywp++dY4lmzUGBNPKdTj1o1WCi065JVwOB8aERKIMD5XtmYDq

ahGhOhFXb904Av3oeQow706FO5HL1hjvetD93yqM8guM31X242xoJoNBWEQFquAmWeaJLqQIyn7H
G3XrmrtEOkrTAe3V3rrKA6gh8mYUFmvWhoIBpcHgKdQYRlKGj0vyl+ABFeRZYTWR44cIQML52mKg
ghlwVqCx0foGzWAhkXlFwlv8+cIFtedd97JcBWzWDAbZuCBKzNIRgyVsLy6kZVIsErUqL1FbCKHd
1+cX/KMJ11oVvIJ5Qq3cvtk/cShRY1ADze+F3YnmYbyIO1op1Def0FqwGOGpRXByeecywV9hh7Z
XEaRRyE3DSXzk09GE3y2IDsnpsMC4P8SOvf5BYD1GskgVvMhnHPPLPPwgZEr+y55I150KISbVjpx
m8MWsoqgTxTg9RyWy0AtE51r78gjfUtb77tf+r1uq/GBSaa664Nxxk1kmeYzaGwOIU8cBQie5RfPc
NHFqPWzMvQsfX5uYbO+TYXbKX/plQWMglqlwRwmJlDEQ0aoi0BDTuoAZTKk+yPew2uLlG5CBw4cC
KWQyEuEaCP4GMsmIEmwzJP1BDACKhhdzrU7lTD70R3fLqdmn1PuD8FB4FQPk1NY7WouCqkHpftTKM
LDLaslad07JLYZiPwyKRhqXWkNWlzxPXRpmZGFnQqAo9/eGsqGeZUSerrunIG5E0tD3YR3V18fFp
HOGO38ism+9YJuXdVXF1YKGSaYSasRSUkXLQTVciyz8vXXxmCQqjdorO1jdOqIkHIGBaQ5Sb4qPI
/WeJefYk11kuDTkFpDX6ypCV08o8URF1ata6tK44Iz1p37qx6LBIP1HWVE0bRciLzMvieZYyNpHm
FXCi6aSrKwLQRIemkZt4OemSlWrx0HX1cgDVue0NjDbgbVnsQhk/qEOm6Jef0NXj+fCMgyKy06bo
Cy8YdHvK664QtUm4lI5r1Ef8/mkIoZ6/JEz6jlo5ild0j2PuTfvec97ArvINgMFlxrB8K0TxxhKx
haifKgHhxieuXQAnmh1bY3HsTA/z4VIj5+UQtyJ12giz5dyKEV4jh4/0mNO6lqnNn4uzpHKjLiHY
6O+BpqXm5lv1E+SccgnqZ2aezVXT9oswoqMIm4DVRoGo15N62fKHXoxkxH9GVnKTJQpAmgQ/KAH1j
XNT4ddeVza8czi0L9vVy/w9qkYFvg4MJJ6ls6gIUYt1HVUbCyospwvPWF34hTvWKFvRlHkieUlnq
zdHgiK79kpmFUPX2esYi3I1la+ZE+UjCcKHVrkvh5RICuL1lJoBwo/ERj7koUYq54R35nHsPcyrc
BGJF0s7R+TL1CleesX3adhVCe+M4WNoY2gA6Lcwm9AtmCYN9I73vk82NtdL4TAPcFCUu+8FyXxYh
w3PrpdtTudygbFJ7e5tsUwDBBrXwRQ14sqwKssdk0I7U7I88ilwWVj4CBdr3iAiXw4zDSL11DGbE
YoAs0aBgGKobx1Sii8DMXPVrCTKQ9tcVRKVTe/1RMgtP+U9QtmJ9M9apXNHm/HpDbOS3KI6TgWJE
/TEElXKN1hDkS8pSdionHQuSGelBwdjTqIh9lIUJedOaL/wjPWeex5I3cls0j6gs9vPg7xIn4X6J
AhnVbmrWmyOwoIsYfg8hHl4P+UM97lrluOSN2qfQZQ1aMTmoevCjLSkGeYq11GpvlYsOHgUZltRe
assl8ZNBbf58OHD7PgZJbjk56p4aiRzSN5c5ylHpiRpuq94lIZ96uSy3PORjwS1f+Sp1lFjXq9sz
VIVPgvrRDOFzdOZRiIAXGDQEObWhoIJ32lTG0K+uSkwdHVRcKkGvpTj8zqldMaURkmzTskgK/RaQ
7pmlMQM/7c+F9aFG/P+auiah5y/mv8bgwm9pEE+VPwmJPEQH5fGFXvRtGrHi/hNOq928lTI/OoTQ
0koDuGqD9Zxzf2GkviNw+b7BAVEkw7ytdsoKfuAq6lTXh12dWECTLH895hea7o5ZMcgTGXY1ghlq
7wyvsBAPCEASJKgxq7lFvGLQskReSi5RAokidLT6l4ErRppqGWhgtcChdrkirpRvsrBAJQ9chMkoU
mnTSimMBgWiDnALukNUwa/Vw0ITSUMjdeyJ53jPP/mTP6aiPkkenlppA5vs56qRGYUJixdhcE3zb
qQE9vyJE89TywnGqp9fVlJHhQfNvNHnllGubJ6opmURIUuOsPGz+bkut8YsH2hPwUjU+Hw7otY6+
KUVSi10Sdk8HEXWRXSSB9aS7YuGHheWesRZUD0k/zgtynwW9y9TqqdNcKfb+u82PgMITpV+SnDMR
08VNpNJsAa0Px/uYirJJ2lolsjrLzSTldcmPY60qx/0p3QKV6EKOYNNiOpaslaUWb4ksw03f5sxi
meZSGihS3whX0LoSxqi9xwI/0JIW/Hg54a/WkwvhoS5Avul0oXCdijb8UVCbqwTySdLAEY0NBYbb
FaG+RbsymGZg+WITNZWz8of//Ef89h6qbZ4NYKhZmGMhAE4VkeFt6PH8BPSrPaIzWlftZ1GMT87x
0iCqoqV0LacSVppso4bOkIEJTRRwC33rDIYB8PsQe4jHaY4Mj0545H5iJxfZSRpaYMMqHNRdyx+9
hgnobtKNXyzkCbpta4OdE6DBGnh6YvmWVPfh2o19iwb1BtsEQ13UHnlCk21Co/+qZzL9OSSV11C
09hTQEENZOhur91ZNmGbU0Y56ZXF4yxuovj1krRwCapdTDNKcxYrQhOz5HEYWZnSXWLwrWwKUzXp
+yoSSrF+9C6lXpwSZKhXObcx7DI9HA4bvWzaluYlWmqryPN0C2fSy65uOK91TFEFXYx66DuWhUcs
3odEeAPfUhaezuhc8N4v1bwD721RRzYUyq1GfmZNPQG1+7vSdbZ9epMUwE2usAczZMpZHB5HDsu
vWx8CO9B8jziIpmqvJobWLIzcFtRjmq3jS5l4IsJ4rWJV6TF+FwnHPYVhEMNfPEf0+GkEoDeRYP9
dJat0/kkVrxZajlCXuU26cnZRmlpzn66hBqBuF9p3Qe2z1JK4eb9ws22GSAZxHXoJk3Ihi5ZcXR
Rby15JgY8oRGecMVYQM6heksbY7nX/qjG6R8ipxLZRvqqGpkQ74eyygLi7JXA/vffLPVhQ9ciB5
G7lgAjex0LbWlCen25vNR8KbKVKV825xPhqTlKVsjSu6NEjF6vRJFaG0PAMC0UbuSs0Rh+KY5bN3
Xffrc0KHi2HMfchGyEUmMLlJOWXfarYkMA7rioX4nhPnDopU86j6siOrkXhCAGel6XKVB1XTOIR
BRA095fGrdnYVnzub3jgAO09ihVqu0zCgRFRYwfTJGzUnY0YpqShOaKnLXiNJRqzDufq9ZhPb85S
1juehZqkIXPeckFz/Ny0lvkJ8j5GTg1Ke9ZNceMLOY2HjX5595AcX08/1HorVRh8JoPs6MS5EIva
856f6OysfP4BxeUsd76htdFr73+VW900/yfg9VhpQDL14aUBTHTxQTMPApsCC5rrrm+pmwWzg07x
DRyonojeFg8gF9CrCvIVZrOTlyEsCurNDHnXgLtmo8t/z13toqFPWWXjjY5X3rppQxT4UmNOMfdH
XouacVrxPS2mKkKr9rzOkAGAMXeaMPQJ3iNQuCSkuEvmrz9qMa+ns+pUyfiEJqemtI6ZKgpKhoK4
1ZxORLSy8yLY9hm9CmhyS0ChqtECLqKNWakYV/g6GLzaKaYtSjcrAtHwXuUUeWvaLU9kuliKwMk
403bDvaI8iYR4MHptSK9/KRIk5MCSKWZy7G0k1SclAKkqViaE8s1LOK3zrm1UI+8ZqKnxkTC3k
CoP+Bydrigg1BeMsd52+80Pdnrd424xfMTtuFNxVklQ1ZDNAABc8EasOVfkCdOaG6VeDaEfPJ+WH
PLQ1qWdmNzCGajHxm2DRx34sYLaxZ8NXVDzcvEQ8psbkZVeuQhesxhSzjsX08j96IkjR46xWwazb
CwvDz23GBs50NGKnOTmfr+ysir33XefL/so9U9FvdIALlVJDxRXkyKEpZyJ5NXicX4AfGBMaHuDs
Hi9VkkQYHO3o5I8HWri1s0RuamCIoUnPmDhc/CxkQhITOGz/AW4ZOJueeTXuyc/WA3YZHeY3rAGn
Sv3QLIhgjynpQcU3jOmimgMvC0QRBVBCp9Xk11lav2Jz0OjqDICwwNviZbeikiCXIttGCX67DGLR
NeAAYmq2ZyECOSWFP08nExe0MgxXHGIWxj/Tt/52/e1ut1b8aQKJfPNEFuyCQaah6veb2g10ehO
p3N9IVUAktDEJ3uZaWbEDYXUZijwrICB+dqQzplRHyiz9EM1VKFb6EaCoPzqAQTFa4EKZihzpsK3

fHc8Fk8CLR4YMkr36sSInR+ScEDjbKmiCS2ZeURFxx+vNfa0WjAi4+hnqg7c16CSn5MBQgaPP+ss
mkxTNVa6/L6iswvLrCHVo2iLNPMQ9Cw7h+lq95Wy7IP1YJPaZGmMfnZYOMQ6KRR+I9gTIzEgjdK
56vG8lQp40ahtZtLSeVMhD10ioSyBNMGxZUE1fb/nAsRpgpdF6QAYG2KVWjFst3q22Qeh8HYdp5l
TSh7W9JUIWwI4y9dE9UkWKVEBpHlVJgNGRuf1bv+j031pPPH3+nM7omZVDIti98HStW4KfwLVCS
5kHnY1KjczfiDgyT1cvb4hR2ST2YxRqAZQCM4nQvh+HAI+bxFJpq8o8saYMo+MKcKcqFBLqmNWdN
1ATfd3WygR4r+mZSZmcmB2qCdqEM5teluVxoKohbP3whz9MT1j1maKjRlFFdYmZn6IMFUgdic+Hh
9k6x08+L+PNEZkYm1QedFxVZhCS67MUx1LzQ4211xRG0CBRYhAWvY2YKAXJAOANXNTQIYhDckWtN
kS/zKVsdTcNTVTMN5pk2ZCRhWtYlmzK7qGiMmNIPBcax2R0P0QtWZo04RlUshCdXsBIBRtwosaoE
ZgpElqtW0Ljg5H7yzEZ8XCdNi5C+mYpkK7POJhXdVK6P5f+eW+sP/bXfuTV/bS4WJN9L+UILxkNC
5KBqokQKEeCnvnnpbHlVH6nwzyuQwvAnV+ZMt5MGNUNcbFvwBrbQU8TlOMHqTS9R3/J4zhX6thJDN
ZR5AVFC95bQJTI/P09Di3w4GfshSGzxK8rGAwtB7733X11ZPTPkPUlehnXUenlcia97hs6YuBIiu
+sDkGrQ68oilP9dHhmUHga5RzTz0JNKBQV4PG349BuZoe/DfGk9rp776uqkOnQBmb5xbsqH1nnjq
4wVvWXGa5FZlKjYWhEPgXwmsEYjTyQwnQx4DAaQWDEWhufWf0yyCfAMr2SZFrkfQiahUCBNlbaG
wpRI3MMaaVuXg7yChuMiasFj1yqY1a1X83DmubVeW+sm9sb74bYMYdFeQkV9q4GsbsPIy2WoLAu0l
qyzRQ2yFC8D68U4u3gei2WPodieGkWWByU7k28xbzBa94yYotR8DdIleSlNagG2hseFJ9XHoAHav
MwLpVwiDmA2w0ZrXOTLAzh+hMA1L4saxiqKLqzcpQV33PEBffz2dT9jK5lsiqLa+ZBWsc2+YTO65
wa39rZwzeXgoWMed8LmY3ROUSG3JDSL93UkqT92at9G1Xx8uOsIqS76aDnw2eWqeLCWHHI+CXlxV
J2v47tf6mxqL0W6DUxkKuS9b+JLIBkJExKiK2sSJ70RvOM8L11FUTmIywgrifiIw9fBI8+ttrA20
IG41EN16uOVLuzW1hrbKVIQsAzrvqqFCQMLUaIoc12/Nur15vntWX/wB95ykQvvLlYtooic3apI1
cHy2JVIXuDEsSLQ4DKPgpYizJorZGUnA3NTvC9yRF3o+jfKxCKv6XlzlI+SyId5vCqQVf00vXroY
dRcuHxJwXdd6GVFBfAYEv/S3y8dOERwaXxkVAXeqC4Y+5YpkU67RwOE6NlnHv20nDp9vPTQFZlLT
laL6y8QG4emsNUeTSYTrz9z5gy9+vJYmObFRUkeSTNveUjzPGBnuW8z44AKT35AWGqgjk1rs1wVG
wKGNPC8q4oocHNMA6t+7mrqgbQ41E10AFRjZpchaq0eiP7ihdsthAw19cxrK6WVWrcn68Nzx1E2V
KpCKmX6XCqUV4bbRzi5mlTa7WKqGRYhZC+G00HcR0CWS2pKNVybjkPZUUNrw1/w3YDIPy0S06+Mt
dXZ+4C7gFMFu0gS38EvoTex8PzKIEEZ18PNrIahyueNwtgDvB5hGIniPuwkl9S30Gmeop0cGOrUt
0K5DY6KhpE7vRMe7bWZm3nJSWVolkVhfmhRQRNjr8VkwkBMNjifqrqftaCMWKKDpdIA3v+OO+6Uf
i/17Jw+vWmjloR+UtvUkEPmYsoZaTmGwjdxr6ysMCSdnZ7RRSL6utgrG2Jh6wjGmE004Vwk8zS/g
hum5fs+rq+EuFnQyLISCTqAKRXraYJ935xu7KPY59a8dnEUhizHceHrropZWPg4LFSnFENGy1kp0
q6YRkTuN9MEv2EO0UclC+MvrNUxDXlKmbSx5p6GGPmm9aLyczAZa+wwPFOMm4Y2SgRpl7g2xGQq5
WlprE+ft8b6A29906ILaS/Pcwn5YDlGz7YbH4VBY+igNaGkx6SGSmlo4CSUuTaQ/9W2pCo1IRt2
EYXTOF/CGdz3oqUw3jDrU2jIRqjeqzSWCQepiUWck7+6huzLWw/dOhIgfDgvdIKWxx/Lww5+Up
59+mh7JaqnWp0SKDbFG+ca21jESidKWZNSofXr9NrMU+ve63I+eIVIDtIBjtwlELwpb0JGiBrCR
yW513wIrOFpSVIAMo/36XXb7C0mljA6wWiGERI6YsIkrKhUELumLxgqDtgmb8HnKsGO51tVTAS
jiZGa7EYYq9z0J1Xfl+W1YDBmWpadDFsnp6pf5ZnVRXNtbtJT6ShQb0PIiCZ0NlPuNV55Xpd/Z7G
5/pP6/nzuf+89ZY2930Pe4CLCK0SSsnoaFHLchHVj0cRcsa9SEV/cJ3Qxh5YeC1eMXrxdqOGhQOo
oIgCHdvXxPDDW42at4DRuxlLHzdjVzXvGz1stDT6mbokRWP9GmEUNIareRTzX910WXUXTIgA58Jt
DeowBfaa4vj/9jHHmCEUCLJMgToKDhd4gFuZasjEQQsbj3W11blkMHDpc8a+RhEnrCSRpGGkgd
amIpnmDyr1GLkA+NnLXyvEWrkN6gWDACUqxdJGx5kiZv3thspiN7FJBRX0Duve+uilFYbRFdR2Ex
o28klfGJUX0XF2sIdlnOtwsTHdPPeMoOofzHVQRc4/GV4cyGzmnQrzgRlCPy+FaHtcoFurywAWu8
oQtSuing16c5I+c18b61re8cdQtzstNsiXzHTXGkTX0tfS6WUA+xZA9IL15FNrjkNoQJmNtGkbyY
okn++dRaELH+2g3yMB7WN+i5ZzP5OU7cpt4nZe9shb+BQVSzj9JAthk4bjdVb3JEA/t0ECqQZwHE
YEC2ZEyk8bGR0KopRrGEP9+Sp588knN8zyohMVhvakG2qBOrJzfeqjtGdUPRnDi9ClzmJtnCIy50
eblygWvnS+UZpHIS7RgTpClSG39bItUgQPFGTR8V5+grzem35lyHSTPajm+bT2raJnkSfw49+sD
deioaFYqiLh9Y0y8Z08lWPOFUUvfdOD5MM9xkR+Sa6Iw7BmQ4S5xnx4W+UXE5vIPEXURwyRDbvye
TVxERmOonBfiXsUeaUUV6Y31s9bFNk5HVzcmHujoyPL56WxOq/60/CquCZEO/3gJBgd29QqDJWQe
Pu8ohaVQ6ViLwFp4RkM1Tlk3+9ogzT0blp/J8kRqCt6aL3hpTmVKSXaqxiPnzrXixnVjeULtmIlw
yPqw/j6imZTJTy2G4fpcJBMMQkRYyCB7YNj29vtSFyvyT3orClUd4qhrXV9FCVlMOzycTyUTkCGF
Ke4uXWW4f7s9FzKZQbY6ePWCb0vTBvy9UrWIOEje5IvUETYYZPSTM/MjJWD42Ixx009jNM1UHW
4ezFITOWjYhOimGm82ViVaEYoZhFWH6XNBXULJkNVmPZNfioZa1ktmkLCJGD7nxxHWGjaU61alz
VeeNITZojpySXEjztQ7s4TGNKkfrot4R6HXc+BDdXLBWE3ihVIREzMzM53zzljf/JbbE3cit5lig
XJzc6+o74Ek8fQuu0B5FIrhdsdcpoq0Uyh224U8GSA2KzBOoKwJlWXUOSeFbowoaP4wdRgvlQA11l
bhgRfzYS2HaRTZ9JVOE4G6ap5VWOZ8LSRZKIkVetvlnTky7P+5zqjk3CJ8fWa8qjLa125LPfeYzX
PwWIbB+nGndlIvEj65g+FtRL9TQqkfPsb29SdUHY8i0+wMaQZaX4Jwee0YVfIJsCa6L11ULAnK4v
ql6X48BKDFA2xXt+M8NgY3IruLlHlyJch5dH/ai77ah7XWJbgRZmHqeqUHy3hXDUwRswrlnHenG7
fEMU6uIPHnFRzgs1PjGcs7iyZRoUc0/y1A1H6qVF8UgzNo18j527CLPA5nGvG5uFYnQJyTDTQE2s
MwIE8BUGrV6/+JjF+fnnbH2er2fcVd3kb2bmXFOy/AqowxL37M7ywlqRh20uqF11NiOm3lpT3qGL
K8gyL5zpbLDM0x0fwpCAOVhogbDwyG2Ua6yKxypIEUg9ds80WC4WRbGBA4JZtsksahsCrCFNDc/4

```
469wfKKDdm14b02W5bN5a2WotZWziq0Bc7YPlU9pbxQcCZo5rr3OX78OZlFWGLpxMS180EWQsbcz
+QxUArnWPM3Aj12/hzhrWIEnn4f5rUasyrlg8Kqhlp6xJJ+aTlo4vWcGolmuN6ss2daxonjsrcYu
SA2EhNUCYUYAl9+FEih9EJcYzv+OiKmSreMXtdB4F/bZm0loKqCptVls9wAxtPQM3nqlSIIElW6f
rStMA8oc9AtRqQFWFJ6X9aFhZolrsPo6PjgN3/r3xTnnbFy0nOWN4m49XtKBfNAkIEL2i0y0HDYX
5Bzux/iKHsBxG6oo4UxNOJwg6tqCwXlOEJeldTC0GUJFzsOeSqL6IWG4mmk4RXD4KIY8TrOQ7uTN
boTzSxeWD+DZ028qr7mTsNqE8hh77vvfpIL8HHoBjLvpeSHwdDMGG4eJkfiz2ltbY2h9uTEmC5if
05QyI8qotbYEDPfQ6qRjlu8IJ+IV4FIYiYK26RM55eq+dRjKkLZpniBbKsNps4q0+zqDJ+ttOd3Au
uFhWWV5ru+nwhmbiR6RLXmZNxzdWIZpxBDeTwc0YoqOjswrOaOE8RV5lcASq8czKRdqdTUNF8mH0
pnqNMFAcIgbXvU/1k3Eh3WhGaAiSIBgB2uv2WzunXddN294wy0Xux07eFCU3Q+hiwU7makRIJTnl
R9sOkRD6uaRqvsZ8dxCDLSFqbJAWc+0KXG4fn2O6it3NrvIlPukmHQ21F0TiBG5D3URAprmji+dm
AZtaDh/gTxpddK2LrrJ6alQgCeBo8gDMR8G8vAnHpKNjbNhhku/KPtWjYxv+auVpGpeigZG0+m0Z
Le1Iwc8S4rhZVoE4I1jGE3flzKdyLtU5zeJMt8lFJXX2o/jKic8VaIKdx1Y+8XokYpYXeSNOW6tY
zElW2LfSsdCPR8SDNPxIfE5Df15FDSJFRgqQipknopcXKDBHrXXEXE654jkGk/IsL9RYE2PIegIJ
4mXbK2pPrBvVRSCgGUTVefwZOXcVh+1WbpmzCTq3CVxuIym5ar6wzV6VWwLrD23TT39Z7Wp21+Us
boLdoc74SmyfAqrWUog1GOREfTh0il3LyqFBONSj6TusnwEV8yGUGqVZXDihDxjaAQNJfsvIRJn
XoAhwLZvmRiIt+6cCvUMFEPFSUVzmgcleFnJEMlhCqBw4wSYI8Bvr12jxxhzlH1Rft77ruXdd1Tn
iDripzcGkPCeq3uxls0Qh3aPCcMcXl5WQ4fPBB2cxAQPGuSebFqO1lpoySXU/AMn1Gz9/N13xQ5f
Z30PhiUjc4wI8PiLztnJKgm6KJNFIOoJZUOQdjk6ipJg95X9dAwNpuuZx051FbCdcd5pv0AphbeI
HEatVipgiiHN6yklWpkhPVTnTNbCnhr2E9GkTe6qJ6qWogfhe0esM9XiWoX/TJq8LX4KNBJAbbp3
1MBw9f3LT+1EDhx6xdGcmfjvOqO23R/67wy1jfd/kYUto7lRVmvtEVGkSy/ywLMYC+ldcSAYeMnY
Zf8Tm2MzpNKXgBjySSIZzMD9kN3oRtrU9IisKOyInCESZMy3GQrXmUwcFHpqrGBQibsbGp2WVoJh
yqIoge07EqGOiOpgRE9Kwvm7rPbnV2Gqi1ntJNTY/Lvr3yNxmbnaxMDcC2I/ia6U5O5U4tsYKmWj
txxLZ9ZlSnJSZCHTVY2Lj+9nOEK9ec94prQ8AdQlIj8uEV6WIirqdA596JEhmbBxBVKJ7GCTB9aF
Yc7F7TlkkF7nvnUwpQSLd1Fih4AJFETx4Jn/Z6PFnwph17el1R47TFxzkUcvYE2jsdld1JNtddkk
OpGk3iwbk288Yg73yggsyDBpJ4hxU3JHVcjGVUwSnyPrp8+yCokDlPkQSJ3Xk1VD03q5biSghUMC
Wr7lQjLhngzvUZrZq0JY+65VPDr55Wx9tPe2wdFvlhkurvYBVPjaLoF09PdDeERRKt9iEtQJSstg
ABHLAGYUINPqfOgWFMF3PFhUy0pgpejHGmiuyJC7Toc3AAk7HiYEG6epyjV+qKK0ZIGgIXlgSMTB
Deggp1VXtGYzVK9+e43kE+BggFOCI+8Zq1O5H7wwYfIGdbP0WNBd1ApfdLgaEvoGVE90DOeEIIm0d
1sEeg4ePOw3oYFolaEfhhmzrOLV5hmqIdfG5av5Tpe4pANaCF0lDGiEI77UlnMzURplhZkppR6RD
dqiTAzvTx7IF+UG4OfnRv6vc/WGpq4fRaVEDgNr686BoBvEBuojkvVWwL2aw2EJNDgN0G3gmXo7e
lE/8gJCdNKzvllKK4aASs0ldMgj/R3wHTzokPqAAUAFT04Oomu+z5gUAsUfHulgaVieggMAgWjEbT
Spt7o6ONrvnlbG60POfuzNpApHTRdCh3g/xmaLkXhrjROti/XABrNivzKN6yF0093ULd9D3gEEqR
WC8GPURCQhnCEkRdhZ+fqsvqmrxm8GY766Ih8TPAm3Qz2EZeJZULXMaQvehiedaC0z9pDa8fmxyI
lDYWNZx/23t7MqnP/2pgHoavVAL6FlQxXfFBxxbSk1hAz2WV8/IkaMXKTvHkygI1ORam4y9CLXkf
nFGeRAL87PxPE0y8vlapJPoQq+ons/GxoYP8WohnLaiv0hlVIhJflbGXSI0xSaUBsQ49rVtExIf8
AgyyULdWHxVgDk8ucMSNLcQtiMq6BdKMFHQ0eROPcWwaLiw0yiYWRilUeQVTWOSHV/zj8MozUFFJ
UKCF2XK4sdhAGwLM29qcWjER0qV1JueSgklEDuuiOUqPR/e59ZDD32i+LPa1f8jwAAZs7UsKZQag
wAAAABJR5ErkJggg==',
    height: 150, width: 180, top: 62, left: 50}}],
    { index: 2, value: 'Gender' },
    { index: 3, value: ': Male' }
  ],
  {
    index: 4,
    cells: [
      { index: 2, value: 'Contact Preference' },
      { index: 3, value: ': Email' }
    ]
  },
  {
    index: 5,
    cells: [
      { index: 2, value: 'Email' },
      { index: 3, value: ': mark@gmail.com' }
    ]
  },
],
```



```

{
  index: 6,
  cells: [
    { index: 2, value: 'Date of Birth' },
    { index: 3, value: ': Jan 3, 1985' }
  ]
},
{
  index: 7,
  cells: [
    { index: 2, value: 'Department' },
    { index: 3, value: ': IT' }
  ]
},
{
  index: 8,
  height: 40,
  cells: [
    { index: 2, value: 'IsActive', style: { verticalAlign:
'top' } },
    { index: 3, value: ': True', style: { verticalAlign:
'top' } }
  ]
},
{
  index: 10,
  cells: [
    { index: 1, value: 'Mary' }
  ], height: 30
}, {
  index: 11,
  height: 40,
  cells: [
    { index: 2, value: 'Id', style: { verticalAlign:
'bottom' } },
    { index: 3, value: ': 1002', style: { verticalAlign:
'bottom' } }
  ]
}, {
  index: 12,
  cells: [
    { index: 1, image: [{src:
'data:image/png;base64,iVBORw0KGgoAAAANSUheUgAAAOsAAADSCAYAAACrfzewAAAAGXRFW
HRTb2Z0d2FyZQBZG9iZSBjbWFnZVJlYWRS5cc1lPAAADDF0RVh0QUxUVGFuAEJlc2luZXNzIGNvb
mNlcHQgd2l0aCB5b3VuZyBidXNpbmVzcyBnaXJsIOhbh/oAAAA6aVRYdERlc2NyaXB0aW9uAAAAA
ABCdXNpbmVzcyBjb25jZXB0IHdpdGggeW91bmcmcYnVzaW5lc3MgZ21ybCCprqv/AADyDUlEQVR42
uy9CbBseV4W+Dv7kvtY97e/qq7qqquqprt6pWwFAQNxYVHEcUHEGZUhcEGdcTBicNwGBQ2JAMNw0
HFGHRxHR3AmJrBRERVZG16ofZXb7nv3ps398yznzPf9zt5X5VNd9MNMJVR35anIuvfdzDx5zsnsz/
X/fb/t+RlVVstle/+07/vjvkCD3xPc88VxHbMsSyzTFMEQfn3LDd8LvpSpLKctCijyXoiJ097Iq6
+ek1Jca+M3ADg3DFIufYTti6ucYeK3I1/wP362v+2ff+WeNsD0w2sN9Y+fousmfre7AcP3QsB33U
xxKvf/P9LlPsBV45Hzb9u745W3GFqyv//YX/puvFBcA9V8DVse28bDEVGAZnxqwrXNYv/rPfvf2y
/gc2sztJXh9t2/52i+THMDK8kLSNMMj1TTLJce/8wLguwDir8Ia+b1/+Ru2X8jn0GZvL8Hrs33zV
3+BmJat1s3YWM8LK0r79+pmKSU2N3bx02eR220L1u32K97++Fc+IwaoJzFpbJyzX+xwfNyF/zUG7
A//P/+HsVoupMwyUGdTMvx0bBeWH0dvVvKbf/c3bv2jrc/6OQTSr3gvPcYacOonmrCuDCTBwsI/t
WBpbfigruOod+t5LvXPPBvx9lY2E/kw/4yfdav/Qt//xcd4z/+tj9m2I2ukZeladuWuZ5HlnRyZ
mN/FnxbyzBKC6sLXF2zSuOodH0vdxwPTxdg7FXh+mFuu045PLhcXnv86eqxpz+wwWG2YP3s2v7EV
75PLq7d6wxYEzsuP02wmqBreQFWqM8A6nz8tL3OrpGliQ2/OXAdr5FmSS9P0n5epG340YFhOW5VZ

```

J5RlR5e74C+VzjOHB8S5XmS4fNSYHmFz5o2291zz/Ompu8vzDyPWnsH6dW3va84vP5YGYTN7Q20p
cFvYKB+1Xs1/8L/NH0hleYjCC5y2/KCB+e/iAR/whv7l6DEfKrc/B7g4W/2N9w85+AYRthlgIN5i
+16PTdoNePlwjTzwmoeOz2ny6u+EzZL29hLkmq31eyYludYeZZ6VpE7hulbplFgsakKw7RL27QK2
7UKxwnWRZbOKqM8sQz7xAmap14YnhiVee/sF37y+Py1D58NLz202rvxWBq0BhVXju3dsQXrG2L7Y
7/96Y313Dinnz5gf8mbGICl14s3G6Wh068ugOrgcYDHU3g8aTtuC4/uejW75fnBfhKtRjieKY6hC
WPZCbX315AyDH3PhMmFpZTA9hoNvKZVVV4Xdng3jlaNcgHDCV4sZW6k6bla7otgWJalBX+zHLNwb
a+A1Y3wWAG0sVUZM8dzT3q7hy/4zc7z8+MXP7Y4u/1y7+ihs53Lb4nwtwL72YJ2S4N/7bb/9sver
v1LzWE+AOWm0lvJZ0KJDVDi6lNTYl0BngQbftJyvHdjXwP8uw0aXGA1sLM8nwAPPgA2w/E0yiJfY
99NfHhi007btMwGcG7hmpDTiPG+LqzoQbReDZk+siyHSwrOxTaSaGEkSSxJkugjx1IBUOI4XQldT
yzXEteyCs/zxfE9MasqxTGml1nOzUrO8fm3mr3hh9s7uz/T3b304db+zbtH1x9bma5XyLYAYgvWX
+3tj37pk6+mY14fwH4iHxY4s55wXPd3+u3eM1WeH5VFFhpVkrVwH7HPUVHkZ/BV3aIoI4A1gC33j
LL0+beyrHx8qg0Azg3TcG3LflsSpxJna2EpkmEEjSajCXJYnwqebYCEBv6nB634ch0eiKrZSR5G
kuaArjwLXmsXhiIa5vSCJo4uUxZuYUDA4XdwvL8pRf4U9OoXvYbzZ/t9Pd+bofKwz+/+i77vb3L
q9xjcrthbQF66/K9ke+9KlX4fdfBrBdAPaZVrvzRe1W53HLMgamaTTWi+lb4tVcBnuXfrRI4z72d
6cqMpjEvAEQtRipTZN4ty6wyPssvjAAxHW0hHW2xbBMkfJUHNFHZzuyWpzJer4Q12tI006J6zfEx
LGQsCZ4j+W6m4INGhmXkPl8rsdY5LGea7PVAWA9rAalmFhkDFjoClw/Xk4ks9IiaIRRs9Mdtdd5
xuNl093d/Z++Nj7/rQ4VufHj1lumG2p8Ras/8W2b/wtnyflxur8Z/B7fQBrALA+wPAkAPvrhzs7T
wWB/5Yiid7mN1st7KAJq9om/eAEFmuJ3cGqZlaWRLt5lrtFWZhp1u2WWS5JFuMTSGvKVRkSRwt8n
itxvIR1XEOb6+o2OhLiYduOpFkk0XKmVtNxPamb3bpMEQsIGLAC17ADUGELIiwkjROcRyEZfqZFr
myghfcEniM2WEEA0JdYFKbnZzKfnRZ+0F1eUxb9uNVq/bzjOP/m4KFH/93D7/3NLzYH+2tTtoDdg
vV13v7k179LCnh1haZOx1fAGgCsBcC+BYD9QKPZeQTFbb0VH/JEnmV7ZVnargeglLk0m6H02r3nR
YosL0tgCHKzyqt2imNaLKcKNm4uLGcCYfbqg7r4KFOydCXYEXxgV5aLCajtFF802CisbSPswqo6O
DAXPirAmKwUtPxcLBTi2D6Adl+iZC2O18TC4sp6DQCDJqdxLHcrhgAa2Ab0sF70h1YaljmHKA9v
38Pi0SctvuD2aVLV55rhM1/6/ruv3zsA7/1Zw8eeXqGs9/S4i1Yf+Xbt3zN5wv8wRqoFz9fH8AaG
8A+DMA+47jezX63exX0dw/s8EpVmc3Van7IKiITr97ZPZThcPCybVunwOq7l4spALNWMFQ16C2AR
7BmWaqfzd81Co2PybMEVrQlZKOYVX2cJoPKWaiBlyfrMR5Tgf/gjTLyPK/ViLZpa2CJ52J7nsxgL
fOi0oosPk/rzY+ZTkcyX8zEDzrSDfOCViCddijddhN+cUum4zOZnB8X8J+jw6OrrwyH/R8Jgsa/O
HzosR+7+Z7fNGp0d4vt3bYF63+x7Rt+89t/OYC9uL4Bfv1toJpPgB7uNnz/IcBjP0riS1EUtSpaw
PLYOrld8axKcIPP+r1OJytSmY9HoLQJqGahJRDT+RIWMQNEZ2SLE/EBYV1/RA+Z67AZfGELKkGj
8J2H5QVIMR7aYFb7aHSXu4viWcwth4AGWPfLvYZaeMBn4efKT72yX/HaSRZYer5a0oKq0CWVXJyc
oLncqXX9G97va60m77s7++LBQs/A2hn5/ej3nDn5Nr1h3+s2Wx+f6e/80MPvfeLT/duPpVv76otW
H+1AWtsAft9HGCNDWafwnt+G17aanc6R6HnPYp/90F5W/PF4nJh2BKtFqCtvgRhS8Lak0GvLU3fA
4AXalFp8WgJ79w9kcl8JulWb9MOZ+AnQFRkAIupAMwBcIKVH+8BRPB9cRgAIADWHV4FumOcQ6HPw
adUS9rqDSVaz6XKM1nFK0mjVNM4Ftv8Ydk9gHa5XMPssYZ1JZuu2/BSfN7p+Vwi+MXM0/LawB3Hi
YZy48plabUa2O9S5ufHaavdHR0eHv5Et7fzT8NW+K9uPv0lJ5cfff/Wwn4G27Yo4jPYTFgnTVAqY
DdleJvVrixrV8zcvBAwBZ+U9wO8v8F3fRiVxk3LMHp4j1kUuR9FqyEsagJoW5Vh2Y1mh1FVmwdrW
KieLnapjGdzseFPVqDkL7z4vEyma/HDhQxhaWm9CURGU32ysJ6ZqDJhLUNRAUNX4NatMaO01Rgz
kbHWJ1z9UUTI5EGKKuJfa2WEwmbTezHhf85kMSPQJcJf2O4T9n8FOb3YF4oMbn5+eSV6k4Bii0W
UoDvmscrTS45fiBWvaz0UTms5XsDVTysHeiReKyOx3dHb4QJe/a3V3J0cFh8fJP/9C/xvJydvT4+
7eA/TQ361u/9Vu3V+HT3H7ge//OxxOTj/9lU9ckLzf/t4Aavr8RNgadRvgILFELz/1VXvRLy15bd
rDO8qKRJ41PH1TyutvFoWpDZcr52bFSUBP/Pj07lQR0M83remH6juv1SvtkU1i0FazXZDLWnlkFd
1Xq4sH64bLlINDcUwGKa+NloAYzribKDEgBbRQv4wKdZFreiDPuHlcYZhM1AugBoDj83y5aSGwoHA
OtwZ1/WyzmOZw3wNsX1TLABX0Dp9RgYkOLfC6xhq9VKMlh5tbbdHavIM286nfSgKm/6rjtN56MTX
LG4s3dle3NtafDrv/2x3/bOTxZ0slgAgMcRgPlb8Nhttdu7zdC/ahi2j5eDzxp7WZYPl+vFYZGXL
sBoGY6v/meOF7QaTQlct0G6nJ1Ld/dQ0iSXk/uvaCoFUNbIrB80AL61Bn94DHUPraW+ZwEQ2aCvg
efLoNNWi2rC2jLwJEYurWZPWNfIHG9ZbkANWmwA1My7AkT6nka3LS7pMmgwfebZYgYKXahvGsC3H
o/HANQr1LwezMsMHeOT+XkbCQVfOjAo8/rwvrm0u/0pRV6ekx5uk5NsU6Gg/YHr924/g+7vcGPX
3nymdXR2z5/eyNuafDrTEXMC3GN8uMpMe/6ffzldwEAYaPRGPqusxOnxcjz7MtFmb91tY40AcRGk
eU1bQZFJbja3R21YhiP+oJMj4TtjqZDlVnZBaPrteRkUlBStVcT8qdFbjQIzCJRkP7cnaTYXG
L7uOQ6mA7/R9+h7NtXKLgC6MAxVbobjpmgyfXxSM/gY4lrpqKYVfm8CSko4b+N2xKtnr70iUxqDgk
UQ4phZ8ax7vdHJeR6Gx751hTwIsJHfuHuN4SpmszuoySsMVVkrS2lpV5Zb5eu98Ur7ffP7ZcXn9+
tj62I9/BGBNt3fXlga/rtsP/pPvkU/MhY09/O+r+Zvv+7uOY/sAcel57qNJFL01jvM+ANC4yLk24
RsqpduUC7y3klFd/B2+Y7RaamSWOc8YFrSCVbQ9fxN5tvGw1F9uNLsAV6T1xB7AZsJHnc9GAGILv
m2LKVeluFwImKKhNeXBLgC2uijC0bQLBdOyeIH9F+L6vu6/yBP8LdWOBM9tqL/K57hYpfESfjFUK
LDr2njUfrUJ/zYIfZx/ILMpjqPRldlkhsVnhYUhlAWghfPDQmXFAYK1zA3i9WplFMnteHxvuf/IO
7fw9VPFTLaX4DO3rIza/s3v/0n9nWADiPoAwW+nV+F63iGoboK/Aa/WU/PF7PHlejUAziwCjcXwr
BqajO7JClSXgPEVILZaVUZaYeJAhwHU1VgMgMwAmGwgjxRUtCenhLWzBRZKmrBuUjF9IvAxuwCkp

5RXLsYseAZfMgZYF6xaqvH3YJkH+9ckwd/Ox2daWsjszIMBYmRSC9nY7O9Lt7eixcbE4OXlJlvi8O
X53rVz2Di5LujwRE5/fagSaY93f35VG6Oqx9LsNuXzpCDdXLgdHV4XiE6enI8ngc5+N6+opnE8wG
k8ejtLqN5yfjT/v3gs/3/zwB/+3rcjNFqyvI1gtU2/uP/1V76mBC5QBsF9cJ3AkSJN00et2b7q29
b7VYvYW+KuF5YXRajXz+YrAb6lYGi3rHm7kdrsrrjmtJFi2k2epJksxhcWMT5dPCDACPVhFGGv/Ot
DAizzNN0di0lNwnLFuRrtQX7Q96sFzsJINFLbM6pQKAM0DFet/5ZILXptLu7en5nJ+fAYzHmi+lz
8p64tX8RIHugdI220NYy45Mxwx44bj8JkA7kaPLbwGdhr+9nkq5nomVL+XK9Ruy0+tisYkBYk86s
NpSrGR3fwcgTmS5juHrTmWygF/tNyy46c3T0emTk8XyC0ej8Y3pyS1ne4dtwfq6W9bNTwM/H8GjB
atY4uHt7e48ZVTp+wGOHdNtL9brrLc4u9egj0jLl+52xGbUF8j2HBSUM5A5LJwJi7pcTTWgVBq2W
tiCZJdRwODTBCD5ZeVFBCDC/4OPKNpHa4nplFaZhfx0c334kzn8T9Jo+qrsogGFzmDRmM5Zwm9lP
W+I58JWV/Ok09EIf69zuibelyWrOgDFdBDMtt8aqC8bw0LbOGZ+TqsZysH+ZSwapqaL0sVYBv2uX
IVVbcJPHmLhaDZ8jTC3sThpmkmPo5LpdIr9hG6eF8PFYv2e2Wz2ntnopPcz3/dd23tyG2B6vSyr8
dolzpNasSEBcm7u7u68t8jza6kVpiwfXC5OqXMkw+Gu9Pq7WlTrghKGfkMKAIwVQvPFuaR5KXGyl
nJcaMEDG1TyjZRLFuK01aBMBCX6uDYAymzqWtvYGHwimJgj1aAX9kVaDV8Q1rSmzozoMuhU4vcIr
3Mds5/nf1lbrJHiypOT0Zkso6XsDAziOQFAxYWhU183B6BD+KCMHGdxLGtjBgD2lJIfXLohy+mkj
mLjfIBCGLyt3jLHu5kKskHFM5xjifOJcMxN/FvEleon+KlJcm26NJ4Zn00+4gfHrIeMt3falrL+i
rf/8X//YQXsX/re/0DL2gdYdnaGw19/sL/zO6qyupFk+e7o+N7NOI4t3wvl6pXrWuubrOawVmtxg
FZSRM9hOYRoEIfbtKzzJRgvShj5WlqhNikfiKeZWr/IdEpdNE8raRs1Fa6KBI+CuThx4Ue3O1l1hV
fBysdDXeR67YypN/ZCGF0Wlhfw2uTuOkYBkiWEUL3I+mW3SOqKWvTJ9pb8RfOgsjeDL0vImspid1
nHx6Fw6nZb0NRqM9Quv8WEGWqElV4+O5GC3I038odPpaFXTejmV2XQs08WcdN1yw1Yb1v1x00On7
9+51f+Jf/Rt2/tyC9Zf2fZXv/4LHwD2z/++D/gA68MH+7u/CzbqPVLW7M/ns4dnk3GPlrEdenLly
hVtJcvjpVo4lhKyRpe+JYM9k/GJRm/XALJQYQVOXJwstKuFlJolegboZRLF2qNKn7Iq6+J92xQtF
uDrWVThAIxr0Gh+TgngMmdL2kkgx/EKzwdi+a4CEJYMTBrWF1Yuy0u1yGyA195VQHwF33KGz9a2O
3w+a47T9VwaraFY2P96ea5VU2zR47GbTlM7hZgi2t0Fixj04Yu3pRU44uJ4+5227OFvgWuoK8A8M
alwkqRa67xYzd20KI7OxpN3p2n58Pj0xN3ebVuw/rK37/yGL5U/8z3/Wn//K1/3Ba7nem/b39v/c
gDsnWleHm1m00G0XuCmt+TatWtyCHAAdQevJ+plX5/qBlZ3uCL6X34f+cShKCLoMVZtFY/tMzqj
p9sY1ELbYWrGwW0YZ2+Kah0o9XZFEPgNbTCoMeknHVvKkx2nkjQ9LWcSi221gN7ulAkOE7Saz/sa
IqFBRWay7Vcpb4k+kmcaWUSglUMUFxWjdr69qhOH5HUzukygyWmDSa43cbdBrMwa4A2H2xyhjn5
klDiyEsCUNbdvs9acHC7sOX7cPKRlhcuEAtVpFlW15ntpw9MltM37GK570f+0f/8zYvAXrZ779n
T/51fJN3/3/6e9//Q9/id/p9J4cDAZfWVbF+7I0vz6dnPdYJrizeyAPPfyolt95jqtAaneH8FFDW
KYzaTRbkqkdLreltNdrS7+3BR22o37ee161rjPISCARfLT9alpVJmxJCg5VHYZT6LRr8cewA+14qC
MOga0DNDt8FLJrv2uJYbIWrFPyu7Wi+NMbnP7DioLTLZ2RHB9+n1HnkoGodQS/tlLLyhTOOkqUR
jM6fdFtxMoq3kbRfAT/t133L5SRHF66CV/XUMReCh3ptppYLEzpt1uytwnLO2zLpaNrGmzLsHhF0
QpExT+YLJZvn57Prs7Pj7eR4Y+Pl2yLIj719g/+30+Rr/u2/1N//1vf9Fv9Xn/nba5lfAVcvvcla
f7I0lr3tX8UgDo8uqW0MsbN32x1hWWI09070gYNvHTpGqyuV4tyVxmAY0kXlJYHYFgEUYL2aqURg
M1qJDq0EcACK6X/dgGqIltpdw6rlkqdp5OKerKM8rJIooXjoLK+G2iLHMHu+dRNqi2rBT+ZllkpL
BYE+q5lmuvrqzLR6LRNJQtQbpZAstiB72U/rBe24K8utbMn057ZDsBaSgg/VyuxCB4sklA1DCuQP
BpJu7+nPrQWbWDhiJbzTQlJKb3uAD8zbaUrWfwrLUxcMytZryosdq84jvXC8vazydGTn7+9Cbfr4
E9v+71/8R/qz//1v/9dTnewew333XvuiHvzfHz/3XEW+a3mjtbiHu4fKHjGs3Peu6yPgK+51nrc7
rAvnTDUdIrZYAvNaWdBQC632vLE489oftgbnOxmMtqvpD7o/satWWEtgTwqDhIXzRwWpsgUT01j
lvlsdIp1qAS87FpwwRKqCJnGfOnKnJagsiyaw9gNgFOy1WLyU6ZKF4Ak7B+YAE1/FZaV2NFeh2JH
e7AAi/1PBqNASjvTASqxuN70u0MxTbWdf+r31GaHJod+LkAftiVdDXSukq228GnV2Deuv1KzQCS1
RZ0kJFMF7eVNURJ4vquu49zfrwRuv8hnp+TJmy7crZg/fs3f/zn/4DV6Q13AYtnlovlV5Uij4S+7
/d2Ds0cN3prMICFSmSEGzgm++pDugDQ50xlnDfaRCvxhwditWrqulqQJVvPtBZXaWMSsgogsmPFx
79bu33ZGbS0k2W2vCm3jo9lMVtK6nZruuv76lOm9FeLVp1S5l9Ju72gKYvxvQVEFDNA5aivmsSJR
pwpKcp+V1YnZWnGXJQUMSPBPZme34Nfvs9pOteyx/kik8VkJL0dnHq+VDre7OxKvBiBqhvah0u/u
glqa+Fno0iBzp6JX3XBENoK4Hw91pY8duIwp8zzvX9yJgVWNB9+93w2keFwKNPZAoxkYQWDYXu1v
P/oeNK+vthb3MLlJ7Z34JYGf1rb93/7NxpNVjKiVd9Pkq+Ab/XuqsgHYautKvad4b4kq2md5wQwK
lDWVqevhQxsTXOd+ib1bEN9QUZVka+SUV8oXamiYKFpm0rrfdUPBHAZifWxz8AlZdhlrW9d6GBiP
74Ha5izdthTqWwzqzQ5g6UOQHsplMb9NkBsDOYp1Jb56j4EAT4WxmryiKlW1jPywDXfHYKp3pAt
WHxLBvEtXQX709BsaM1ALV7BFDP626aZhs4RznFwoHkkmKzRm0LFsMezjPWGm3H8A3NipNHZGm8
zpYDpsScC2w4qVgAzYL/UG94zjVCHhVFWaz0SmTJHplT9d5dvL8T64vv+MLtzf1rL+0luz1XYBp
rcUafzFuIHe6YeNnu52uzs4B01b10Xt7F6B/2ema9nZu6wpluPbL8r4/EzpbdzAR91+sYB2Gz1B
W3QSQHQz1XxwThR4obCpvpqiJXWBSWDjxrCOh/2G7LTeljuHZ/I6flYPHa6aP9rHYcpcrYNlBqwa
Tb68HenYpkt7bAhoFnc0AAop+M7sGRHwpCyRRUJALOApQ1YoIFFx7AH0sTikIH6epahVrnAcZ2fn
cnuYCgr7DcoAmk0Byop4zuNmsIusGCVGawlzrVYa/UVa5s9HAsDarTmebGSNhY3A9fH8Vry0ksvY
VHAouRZsM4N7ZHFAGdWbQ+wMD0aRfFeGi3ZPLSVgNlGgz/19u/+7rdaZVHtwgK8z3SDx1vdgeu6t
j3YBSBxI1NBQYXGDEtmJ3fVutDfm5/e0RyppmJwI7MA4PjkWEYAL4sRytLWwcr0M9lZxpmu5OVGL

KyytftQrFwxAKScur4AOqNBTVi665f25cbVa7Lb7wL4AuL250BHMq7ACxZGItAXbtum2OFksWcb
F5bfMq/0mFmizrdo2AvGDAAAiM5858RKLYbdNU6skifvjVVFVcpfXAX+Pa1zNED4PKyTi05bhNWs
iGz+7cAxB6etbTQosyWak3r6LEly9EtabR3sAC5cnh4pMfDa1ZqvfiuP8vKyqLhuY1rUZJdTvNkG
xXe0uBPvf3Q3/kW3KtZiBX+KdsPnwGtfr/u6v2wPbBoUSHysF7OcJ+CZsK39BsN6Q53tJ2NdbVs+
CYoaHVxx6qvWiYLaXd3AWpPlrOxRmtJadmpUjFC7NbC2xX8UEaDKx15YyowDAIsY+CmI2HTl3b3dX
em32xT41s4f2aRSCEwtPwTQM3xeo9nXxvJSVffhV3oNrURqBD3sPK07ePC30JqpBlROQOonZlq2y
KCWadf+bVfpg5+kRarpKYuCTGZVS6Di+NlNxBVks718x2/qefDXj02BTBn6+Jvalj3xuBIF49oN
lKzOZuvsNiwmmoiDfi+eR4nzSB8oddp/8L81k/HB098YEUdt7D8JJTDYgKiGgBwbzOM6hHcuIUTN
M3leCRT3GBZtJLOYE+6/V1JFmPNpzLAUwKonpWAZulrGSH7PFfrMQ4r5qMymM55UTtg2I7G4cX0W
V1UQEvGG9+m5bJq0qPWiZHcyli2mOeg3tEU/uRA+j34ss23ygvPPytRZsq9u7fUr4WZU+uaJ2O8B
Z9JuqxSpGvQzT2JpvckCxxvSzs1wtTI+2pZNCWALFRYF9p1SBkZ9TNBZ5kYtLB7z1Qj78GTYGWChw
vua0XZ7X7J4IjEWjNOT12Q42NfPzrKF7s8N+oKLBdBb6hsTsB4WkPNbH5b+1cfk+1ufkOc/9hHZG
Xbl5VduA8BDysZYrWajm+f5tbxIe2VeTGQ7N2cL1k+0/TctqpiciW3pouP51Et2qTI+mZ7dNlsgl6
7UqNrBX04Jl6zaPJKesJlMpoSeu01dKqGDF7R6uAq1aokaSaYe48QtYM//B5zlhW60hb24tF6Q/C
YpKkNJKMc3CIgfWA1M1v9m7XEUJAmgCAN68dl3u3H1ZjIMDGVfkO1trjs4pMHOjKm7hhFhIGNFNp
NHb04WB0V2WCfJlPMPT6ak0OvuymNwR12uqomG/11PR8H6nKys2xQPI7IFld411+Spnyha69WKEh
QiAPbsne/tXVeupAAVeT08kAO11YVmpnkjr7wahNLHIze49K639h+TaI4+J9dyLSrHv3mEAWFemR
hyvr4HR7OE9rygN2Ij1u338lhezCYrXBkhv4g4cFFl0LY3XNkdIMLHfbIXyyFPv12BRjJuxgE9ZA
LTMCKqWkRtq2Z5ZeUppHdDAFqWJO21YoVR5rrTaXZV1WcEq86bXqqGCyg6WxOlC1QURtXYugFOX+
jGIZKYGwPKydhCoAFxDQUXN30uXLkk/LsU1czk95eAp9pQOJc7n4rsdjRBbsGyS134sI858HxuG4
niqqZYGFo3l9LZ0B5d1MT0WXA Vzwp2QjYBrGQw7MkEno6tFbaOqVUm0m33tOKq1d2T1fwc51DJ+
dmJjtzy2bkMgK4kB+22cF28/r72yxp4Pd0BLkyrk+eksXdTDq4dyTF8/cMr11R+dXV+PzC91qHje
JfKLP/ZLVi3Pusv2n7wu75ZG1dwt12pqvKpMo2/IE3iR1bLuZkniciqLXHroCWGnXASg0tIwlykl/
Tdb5t05TEGVx70VjEcjiaJakoXdk8yrmlQfZNSZKGae5QB6IrPpGXzYJQAylfunr8hysdQbtqxYl
G+oAv5kPJP55L7+fXI+kcnZbQAw097akgsC63RxUL7Ljh5P2/FqZYlMc5xskQv8jhgsQ/RDTa8If
Wb4mNrNA3DRz8w5fiMBRW/Wdcy06A4WDPqoPiNawmkAlQbQmo2OLgra/cNAEUCthSAslSS1Zz1yk
7XPLaXi68mxakmxg4eMgRSZc3hWk3viNQbS6e3IrWc/pMdH5RrPduJuq/Fss9f96PT1D0U7j7xna
1m326sbVn4OKA0Ny36oLLL3FEVJaQfTdnwp2ZgNkJ2fncJSxGpR6VMycprBxystR8awKobtaPQ2i
emXdgHgcZ3W8HHDh03VLyKNpTi312hLGxaOBQZUC2w6PXwrjsqEcg6N9prCH9Q4DUDP3lKn2dS8K
kGasFQRvtL2Pvi2iQStPbX8fN2Kwmt+VxeHjA5mmWnTOKVBQwB3MbsjVnioWk2MRPuUEGVPMKi0N
p1TlQ2WksfFjiFoklPNKFD72WwhZ6NjUN5L2M9YQc0hVQP8ezk/U/98PJ2o0uHwRHO3wCYezIf3
ZIAVDvsHUGyP9V0U2f3uizP7wD8Q3n8qc+T2y/fxoKQWK1uv4HLcGTyfgsL5/mb3W/dgvXjwVqUt
mtWl/OseBvujSGoq4e/mVQ2oBq+dr9YBJ0hzXBHp6+1cSHz+VrzmxEC+pqz00WkBb9NG8YNW8JuT
8v3mOapGJ3FDR/Nx1pi6APQHlUkmvDrgJwTWNfZcgrAhWqJCQKTAmiGoX4tAcNCfsvwxQKwFWwAS
+/K4+rXchJcGYbSWIE+M5daMbhj1ur++MrzfIlzoAhaqEUcVDlU/zhj5RW01epqlZL6M0qcvqdJ
+MJFiGjlaasXq/LAg9DRjJn8ehcBoOBVGkpqZnK4vxYusMjTe/ce+Vn655YHMPO/uWa0mOfMUCar
mYS7lwVA59Z4Zj91hA8z9P6arkQE7uyinTRYAv7sErWg8JxXlGzvgXrduP2r7/rj5u+bbSL0njIk
PJGlucD+JVXqDDPcYeUXGGwB7e9tAZXZbaYSpIVWhbImzJPM62tDRxQQjvY1KzXhQtWXij15bCoP
GYDd6k+IgsraPGMwqfF0LCjHkAew2dlCxoDWQwqkUIzL2NWdbqECmmUUzGq1C02GpAq4rHKuPCzS
YHtg11VaeBQZZYejkHPS7LSQnv2oM4nnlYNcTFhrpQ1wD78bQK/hJXlv+nnUnyNqhKk9+ZwV61zr
0uAD7VOMocfdNqS4/Wu3ZE5KHpn94pcfvgcdn73Rb137442IFD1n40FJiwody9HL/8cruNlcUCFs
TpItlyICd/56vWHJcDxj6fHbrvT3zMca2BuawK2YH3tBlA5pWEegpo9nqXpNfiqe2dnZybFtOmns
UghBe2NcFONP/LzSk9JCXWSG242Am13CBoautIE1XUpuULLyKQ/Bc4MqhI6msZwGG/mLbm7vQkuZ
Wo5GahqciPbp6PC3/T7SP6yMlarmAGYni2t5ACqpVFWy9iUKapYN8scAXqKm7mk4yvptFoy7OzJ/
nAg89VclqsIQL2n0qPhZuQFG9k52ZyiaHRNK1D2YrmqVSvyRJzcUd+X4uOdb1/CkJ9ryO27d+tp6
bCQezt7ei5WEMh6fiINGcru4cNy/97L8uILH5a9gysS4pg6g0Pdrw3/dXF+W2ywiEZ/H4bVkgQ50
o6g3u7ex6Rc3mi2uy0sNd4uG/2INNwKX+z/fjfxazMox+GsdFWJbV71rMJ+85vnf7YM0Epu3ih
kxkBCrYas9JlPlut/fpJKSjgZfXXeAP3GUXuArptd/NUCJSbp7yxTYxBKdJYx7JUgqVky5vT0Nkx
F9+DUVf1SwJLPlutAdhCW9gMUMTVeqyUkoUMnueKVj7SjgKsthPUKhJqI0vtM2XNAief+1g4WF1V9
8hmDyqlxwD1L734bd2Eym+pHx2BnjbCoYy1WL+2tgH8bAaSfOyryzJpnm+nJ45Aq23kfLKNRvTY
fnb7b60mj78zy7oeaYW2XQ7Mp2d4W0R6PB1fHZcp6ncujmf/bhZPJpm7g0VdtPCYQ1mVSD5vJo7Q
etv2477vUWenWRJUrZ79/9Pb8qbdhsN3mznL33IqYryGqzYF65n58/E6+V+nGYejJv2m06mZ+qHc
XIardgKFNh2murjKXI2fReWyRejTMT36Wey79TSQBBnx5A6WqWMMjn9HDCqfddkrf10BnxsWGw2i
9mb6XAui/h9T8WxTYDAAeVtwW918TMA+Brwb0Vlmar+z6Yh3KEfWo/MUsUJVZvgwgBrxfpcWnqCj
H4u03R8jsNo1ALhLMRPASj01/HYR4v3mVqoUfe5spWog5ctG0yBfbNS4L0dHYsBe6+VTmmeaT8sF
xDMkkn9S4AzdeJN50zPXhE76KjbyLCBgSr97L/F+WbRDMeaJWUdhlaZrOmZSc41JVVFLM0T+dFG
pd3fvoHy8vv+OJqa1nfhNuH/tnfMHBD9qL5+DcsZ2e/L43i9y6j9e500jZXUaFaQTH8TANWMEqSj

bVraMCFDeMBrEAXlpWWI2Rxg8tZpX06sXVOEzclQUupFWoE806tSoCiNLXqiQlWUlj9QjRgRFmWX
ApOiqTADVmFBJCocj/2YWD/1UZfiQEpfi4tkg3ry7IivoZR5BT7zZIJFo+udgVRt4nuqQ8r7FLT1
6x0wYkWc5nNZjLHApTCB9eRGCx4gN9MX5fRaVYwFZtxkt58bapQtKiI4QXwOUGP53OZzhd15Brnw
Zpin6DH9Wm0O+KfBU3x8DpyFq1K1GDR8axQktWZVjY5Ya2eyEg6SxVtN7xjOt5HcRWWgO+H42j1o
0WevoTdn+BMxp/s+/x1f/Rvfk7e1Fuf1WTLDuxocb4Tr2ePx6v19ThJw+XsPgxoKGk8kTRh3etAh
bMr3LhLUL/F7J4Wy+8OHgcIculh9VnID3+Nf3dwk+qQC1Y6sE3MLLS9jRS4pP9JORS7qoM7paHFB
OZG4V9LDGFB4SszjKSWMWqOba2SYmhDUAWbjeFmPJfKdNRVzSVXhIVMF/9Iotb7YJBgYoY6WWnroc
lZONVcli0a3p4BvUALxY/3Hk5EWbHBgc+HgbwYWGzawswOn2VHqz04jx2ppI8LFYKsu3sP3c5YOq
XMc2VphzACuy5pmCqolezrrlWWH1G3KQcUpLdMcXgcVnsh89IpUeaUpMHby+U0TV7EC7zYWhlm9M
4kWfx041Co9/2Tf57/5zj/84Pcv+Ka/vaXBNyvbT/yDP29kq0UjW0+fSJPoC9I4emy5mA6ADKOiz
CcsjKtN1XP4ZbAK8K/oDLqwToPhnlTxQikr1fw4gY39oc0G86C1yqDSQC6OtVCsDK0m4tVidYDs
3PUMOhyApW7f8Uje7Sz71QKbaV5laaKiJAL97Hv+mrKAau6vulpohYfEBw62JQlbUoOJUKlQ6nG
rVmeIzW1gIkWp09tYBS1or/jm2qnAsVEhn8MksylbpgumWqJTKKEaEUsOrarAC/u4UFIKSGE64Br
bCpDQV1940uYE49WUAL/kGRHRZLULA8r48t7B4qA0yGeZEd8czwRhO8P4IO6lwxOsiy45xjjM8R
pUylE/9ePE/fp/ceP9v/5y4V9/04fD1cmXG60Uny4rrWZocmZYH4+nDVQplAbpGxcLl17FSpMGyEr
GFVOpUChcdg8dqc9gaKSwyytJB6TKwJpqVl8T1FypSulpkCVWtjqaPke3UEFzexwR5XtsUxCAUwk
GJfKbDbS4k3eT2NTTTaSxUKBowY0PF1Do7og9PemM5R6s1qKVhZTg5gSWKzvS/UMabPSiDTakfLs
cQsaFivtbGclvHSjcfk6rVr0mm2VC1Cj4MdRPRLAcxms62F+jxHVlRRPma92ugMYwHwYWGPlt2Qg
6Mr6iIwCFQlucAT0OsJPq/FHkxpMWLN862tc6VdQg78607OVfWNp+f3gnR+/jSeCvHG0HbcPbzOx
GOqWsqf5uOD3/51Wxr82b79y7/xjQZ8INdxnYmqi54qsnw/jtZ9FtKzbI5Bm5R5xGgBK9jUOTSd4
QEOJLWIllpf6wRdVQ9ktLTT6dd6Ryz1gz/GtA5vSP7NAehMo1TLSTN15KV2uJai09c0OFC5tFVrS
QXONBDlqy+nVJ2WV4NQjqabNGrsbXSdYKEdUF4jtxTMuZFqiQdICu1VNVj0QB0lWlIeB4BAkspAE
v3vaL3S0kONNGPhaVKRsTWUU3bxAfh6rtlalhcKrdCmDwQF83DONLDN63Mb1IH2dT0+kAYvYAN3t9
Q5gZSeSdTr1c+Pb0u4f4BquL6bDqwoFc9005Frjya6czTwf9sp64QAXXlKrqnogDrersoxqtPVfY
2jeVEUSb2qwJvHKsGyHsgiflybpYl8vCymC+YFywU/zfNMjapa/mXcWLgzTKZsArl1762Nqha4fP
aVapaKC2wy1/Y0N345Tz0wlUNhwbTtGTTcBFKZXDEaNY+lzqnZolnUO5MZGVft5deCvzuFyU8ule
RlDRcBlvAYpLUSNqddzSVSQjJpK7PSpyrq9jn5jQR/XtjYpHVuS9Uzn4BAkDB5VGWi5TaWJQheEf
D7CMbnS37uswbAcYM5yAwsX6DUobKliZ/Br11MuU9oIYDFS7Xdq/5gCb2UNWo8qi0z19B7TJoYcn
12XY5pgNWPtkWW02rabqjDBgVyWSTnUqEgW5y07bFauF7wE+jsGzafso7PxW18dlPtpbD/47V8vX
/Snmvcl1s/WrchSJwyCmlm0fGe8Hj/qBf00TqJOVDtBXHKj35vCV6XgdQJrRTE5j57rZYm+GHKN
JdYaqQ003CrjZveIlpoW2nlzPqeolH14Kc6n6qDhdXSSrWMqWJvUgWppkuMTYTY0hub3TG09DRQO
v3amlJzFCSBa7puPXiqdOt92AWYzq2ISKte5oZKrbDrJezs6LlZLiO+ZzI7H0tnANpqDETGPwDkE
ACe46PX0tk/kssa28nt01Iu40dmCSj1QJbL87qtD/tm2x2vS1kFtcmjFhUVLqKxh0098TkSCKfYC
FpquTl4i+5FA6CNlyuJctZFT3RB8hs90Gz62+4CF7obr6bMWDzdbPUCnPOP4jvjbKGm1AUSq8/Eu
v6r7/hD8hu/+X/Z+qyfbds//wu/1wiw9Fdl8WQSZd4FKtoYDMvCDWNR/S6L3kHfVJU+KxVkeV5bP
KZphv2BmkNPb/jZgxY3s6wpnrmZNmdrDL1uf7RSuIgOU6a4A2lwpb6qs6lAqhRctWSoVVtr+oxmo
T4cKSzHV7DDRfWcGNDCgkH/19rkZ21dNW/Lfxu21jAymOVrbtdUIbf52St16gd+ox8OpD24rjnZ1
eguaLElq9lIpUaX05Ek0zNpDo5k9/CSzm91b2uxnmvdMpvTudhQxJwC3+yNpWSpXgIm9wBiQ3K4E
KTt9KpDzqf2vbHwMBjHyDjnxbb7R+IGHY2Qa5M+pxOkScPz23Ne1/PTV2R0/6XHsXC+13YD+K3y5
OZh6Nr2ix/mJ/o3gfgD3/6Htpb1sy4MDjPlut7VeDn+dY7d2DccB+5p0mIULnNoguZAzk60tfavR
fekauwnZYtcWTDQcigerARVGzccxMvxwFwCp7NdW4shaCxZ9seWtNqS2nUBgAabchUwYzS3JJCBc
i4GLRYIsKqnelCMoNZ1ow8sam0Zoa10oTAo0MRUKGlq1FjB7pQqP2oYqRb06yKCz+MefLfOr0YsO
ZydaosbQc+8rQ5xNm1ViKDV98NdoVaZ6YA5mJGEvYH02LbHHC4WHSorUoaGPmYaL7QKi4BlgmviI
tbcl+vEghLLOdBjyFZTYSTOWutYIkLXo0rGGhOgbjFzqzTBDECDH+MEM4ddS77f18XiDKC993h/9
yBO49WP44V35ZN34ZSVmUZ88KKuPvjX/mCdR9ta1s+e7Qf++jcYzU6flvQhyzRv2r5PQZbQtOyi0
RlqITxzmdQa4saulAxWiK7mevMHni0xJU740/yy+d19yZN6jqllbCwp0x4bn9GUGmwElKo9sAhCE
yfGBshWbkeIcpUBrSlu3WXjPJhfQ5rsmM6rU9I5F9W1tLmdAHPdWgCNLqFGnO3635xhQ4vrcn4rp
58Hofa9EphqyZJav5izd3T4slCmtCt2gHmB35XT278g2XIqnZ0D6VNDGO9JVpMNe7A0GMUGehchu
ghsJxStnKok6F8HO+DQq0wF2SR3tP6XqZ0yW2uqhsyCfm5B/SdWXZlGkqaxOxvfcBCsKv13EjNfE1N
mjzTVY7x3Gcl/AIDBF99WF+308eH1+Dx1fjCRPvr3jtf+Db/sAWrJ81dMJ1TUCvrlGH13EjNfE1N
slWbT+ccpJakqy0h5OWipFa3nQEb0AB73wtvW5b2r1DDcYYla2Wig3TLkDH+lmXtJZwJNB4hQnGB
7W/1lby1Ms/fdaN1bxQAGRwRjIYznoWq4LV9h+AtpYz9ZSGcoAxUz4KSNBK1iBr7TEWGwaMLKZy7
LocUuNU8KmtjSRpww8kbLZU9ZAVVoxmh+0DHYHBAozRKz8h8fw+LOSuTpJj5RGVCju7e9qsQEPOq
VFbc7ksO2xpsClZnEjKWbO8NjjOJSj3cjmnLRYL4GexhsEod57VQTMVfLM1wKZFFFw2sxSXxZ1yA
WNUfjE5r0dUws/Ny6Jw/OCteZ55eHyBTOf/9VFufpp4hHh8Gfz5g6LIf74GcG7qQ119dgar35Q00
IRpLPMST1I1MG3fTFbHAWjvzDctov6yQdnKQikqKW0S1aWALNA/uHyoliCBZQ2GR3Ln9kfxGk/9t
0olU9qqNEgPlakdSq+oY6XFrkZd6MCHab4a9aUlpW01ocMaSeYNbNWv571FC8S/aypHh1Ix7rTZB

wDvwOqWVlkrI+qoCnw+wJpR+pQwKOVfgQoRVWZu1CLgl7d6qmgYA2TMoxI4HtgC5UKZ94wWx9osX
sIvzzMXrMKEj9nHfnNZp6kCXXO+cAG40GmTAI4rWS1UBcJt7WL/C51CkFA+pjFUoBawzhwineia
GmjgpSJDtDCQmCAwsdhe3C2nI93GFRbLsbamJBGY7btdHfwHbELp/gkBijE491MPeOi/Qgu2xjX7
jnTMEtDKYuUP/BtX6uL5pf8mb+3Besb2181DEYjaQZtZ3LnuT3fb0e2G+RVkTtUfGAZHSt3ArcFK
lxp4JEWzw99abMIXsO6lt5oVGfo9A9UYZ8dKYyO1qNuUr28dYVRVdf7lsZGtbCmykpz+ZsKpFkac
LqIGNOKMn+qVYNGRhaPFoGWuVTwFg8iv/Xg4xr4Fp/0PSw4te9tcsAyrHJGmsnihsWszOvWCHkh
RrQcpvtuqgDN28Uz6WaJSKtjgZ/fP+SjmbE8qWpG2NZSNjZkyYFzs90tbtQLgNtDmBl1t/tCfNcJ
fZNuZqzk5dVD7i7c0noFCznM20EKBSwjCLneN0xLCarpkKxqFaRJK5ns2ExvAerveMEYbleUVYnF
WcnfLnM0w4M7GVcyJ9+zddqbB7g6XKDri+uNkAqxlg9btVDqrmOGKwmzDQIaG0t6xt5e+Hf/mPDd
T3T9Dv+7OzlXbJQ+HfwZbKQFGkRRbVl0o2o8gaEq3EybdZLoXxG/2ZHp6QvSv/xWObv3ijZxU42+H
7JYor+ZdWqoBq7CkvNNye5oNS37AVBlU0jI3KnmR3kDUXJUKSuBaqm3WzkANKg2szeM3jJQxQWAg
SaxNrNXWQ5obL5K0HCmZzi0mXXFtKz0J+3S0b/nDowUqVo37sJm/XM6lUazHmrlU0YUQNZYwTzU6
QAdgI0WMVtPJZ6B5sLahr2ujoHMprFa18qqJ+Rli7k04fdz/AcHOGdYxCj2pt02bq1IQSUL5nlZ/
cRqJvq4vr+jvbS8lizzhHs5sDvepNHonGd52sJ34jIHi/OcZ9HZE47nD9MkvvMaV44jTq7SquLKL
ihzhZxvF2A9T8FOSgtmFSyphEtgWpZd1lF2S37kb32TvP+PfucWrG/EjaLaie9Zhuu0cEN2nKBp5
FXmmEBFFkdmvDhXKusZjloiu3LVH/T9Pe3hJFCHw0tCpUNaYFoJh+Mn2GB0sFEQjVVDhqmNagQWg
01i1D5pbQLqUsI6iFRplJdAntT22Gp9CVRaRHMTpNKxp0qftXZCfT/SQxOWkuWKzM2WBDyopQmAF
bpScEayjgH+K9eckv6uznvFIlQEGrjJVUCtD+ue6oKh6hSdHa3/TUGNrfa+io3Hi7sS9i/BcvbV2
rMmmAuT22jIdHKiM2tIqzP4q+kaAAToudxY2iqIc7EDrZjSCiZSXlJfDtKkd2GIFwpUHenh1NeRl
q8s4j0vbLl0RYubuVW5zf7BiCXEYCFGmmUjsA/OGDnC43E8OGNzgcccrlKGS3UHj+hBCg0AtR313
7EWEqx2HS/A8z/xPf+dvOvrv20L1jfa5mi+r+/hjh14btiqqpWbRUu3cppVtF6YQditbxahMHagE
WDqENHy0EX0OPk7aMq9W8/JcrVWwHmBs9H9tTSXqpTrItKrcp+laiUxwHNxIxobCkYfUdMwVS3kz
QwKg0nqbpW5UmF11ldZB11DjN516qm15VTas2y0nsmel1+rqv6TLIpwq/yQn1CHhuj0VZu1YUU+
PotgET7Xynnwjwso7TwLamKT+XDZDXDSVDjOJd4/JIERQNchIZKsXRbA4Cxqekl5qAFVrrMchUO1
0TqumICBftmWWOynmiuxWQ/LFaTOF3pkCymw7oNU9M8Eto6DpKBPVjizHHcQqjFxnps+wYkoMp9
86sQg6TaP3luNZt0H+KgN+nb4qfMa7ArbpoRFLNMs6mm4qa8Ej5zU3zNqyig7NMreW9Q3pr8IEu
ABpmkaHsE4D0EEXq76xWk4MtSy0lBTHzrKN5bM0qT/cvyInt35ehpcekXg5gzVhZwjcs3SGG+1Q0
y0X81KZgqFWU10kUfTWVesn4KSuXjLMSj9LK51Ie/k7eKnt+vUjwLa19dpIwDBT60mrWKqO3NqS
u3UZyQsWjfnV2vwSJXNWpmi9AAYBs5otlMXBhc+uAe/kvTYZDyqqJUrOK+VKRJaY8qUBvBbcTxJS
skaV6ltQTOL81dKr3pUmbQHewDaWiL8nQtZtCpUFI4VUGRnvDyRdTSBjzysB2GBEazgf6ZRRluGA
ct8dEhh8Lb21urMOZ5rVbYc2zGcVv8+fNjD0iwd26ya+Iw78D0fij3fBBsa4dw58+nB5PoA4LiMcY
VJ3G9I/NbR5X6+xZW2KTOzNYmnK5kLKT/39Pyfv+Nq/uAXRG2kr11OnSfb7yXp5JY/XLcPk92+o9
dLIJqd3m5aW0DEAU4zvyWDnkuQ6TsKXMSnk/Pxc1l1IMhxOduHkaUdTGLUF9TZlNemmKqnUG7ba1
B/WzMuswbxbxby/yqUqTYX0vIr8XQCVT1kJ++HnsVqnn4ZSbm83agLfczL2pwwfgdKJBqKKWUWX6A
pa3curfKzuHDytKjRmX0hwrP99sKaVmAYQ2gmPRYMoIlpGR24W0v2AjPP1X51DVesOvZwqJWkyMA
uuYj3gGF7olYaOL51oqas4AkuOz5nkt0zRTMTZmpY/v35Z+vyeNsA9XpTVJi6RXlYkDcFVYGHcyj
aWVRy3Hb05B8xdpkr1YZukHLdu6h1Pg4B5H+YwhMw2mG5RarpUgzU0w6cKaaupM02ebRfCi1GlRw
d84Wza+axpeowHf7cZydOsagNXI4admpIm0T9rhYj1Qbch1JGKpkp0UCguaOxqkiaJIilgVjTlZ
lNTNxrkpVRvZJ6tdY1gIXqvB/K6kGhw4X1u2g2tzbBIBXmZp2uqSv763QP7ydzE5RSObKcrG4i5
d5ULzHwVVOXZ+j1Yp1+SKjviLWpk2v1KFY3JOVsm8Vn+/bsI01LG6qwbRCLXyuc1cdVmgV6aZez
1RanTM9s5ro53KSAKpn5A6kv9QIztCLDXhRkTEIAGojr8/dqBeZ9eyuqv+bM0KBPrvVuYrFb6T9u
stlhMu9kP7B1Z5Xpj8nRXaI98HJdOZk04Zj5oblLnAuge8704b47y+ydIzzfYltEzjJOa7dkzjVn
2PUT6bCRj0fdhMB1tVSL5C5wedFK10pP/13/4y88w/+1W1RxBtiqwpW+/Wj05cfhq91UOSZb2hDd
KVUL2kEai4DIIPWUCOVqrQAC6Fgx2sm4zPVG+IcmChNNGmvyvdS44sAJZ18YNj4d0aCK3kQYKIVV
7+J7XFibJrKLfWJNdJr1K/KvBh1OkdDTvRlqf5AjSPberXnVbt3ygeLgaaHhsjEeBrQ4Ykz6qoVQ
xx+xc4ivJ9ibByKTKFyGz4qO4bUx2P6FwAk0TXLSosv2OVjOaE2t7PRvdjkiFkEwSNMQYMTLGzME
TNyzQ4kS8sSNz25Vt20v5yNpEgLFrvPV2fi2aWWE87mIxmDqcxGt9lFgdP1fgFIZ+/qK7CM5ziup
WFZ5/j38/F6+VyZx+cAc4XHNZz6Y+1u9ztw6l8E//QZ+KoBfpqmeUF9rRqoG6v6AKhlDVRl/78T
9/zZVuvvhG2dDV3osndA7s1vA6z2auKjAV+tTA2CwIow9fi/k+jm2kxWB3CjVbG2Zb86SMAFPdk
CkRzjIztZVPkQMTf+1ltanQZrKFqA/tKgeBchwnXf1GpK2EK0Kgl5zbpGOPYyXGy8W0ZpdUeV2tTL
CEXtb/GpoDiouDfUYpcWxNbBbPVj97s01Cf1tIIMoPHnCzgUoFQyWnD+LCBWM2h2CHLDNvwPUNN5
XABYvjKN1xpcYAWq7qYgokjFeom06BvSr0oIBLAndfXAm9UVQqyE6Nu9aNB4Xd62rWzWtXmH/gG
NqNpuwMD3BFHClTjge5L9PR8RNFFl/GtWBKBUtdAPL9OyaVUDPoDnb+Ij713HbsHsC6xt89LCDHQ
avzVLPX/auwor+DBRAXIP3PqO8nAeobXfD/TUODY2hmxPdf9g1xLsPSXU7W5w03aJuctulghTejR
ANCap1A0aL5QmJ229D3JKCzWOIkAS0sZJ11nfjzrBYpq8wHVpD+IASqsoxVNwkT/NoCp/NZqelrs
NbYEduzazExvyFhYOicGQaQtJqnqjWK1T9VS7kZTGywY8XStrz6pOpmAAKQ0WRzo0pBQJubSqJiI
/UiF/XJtJBcGFjRRG1S261raeGrshuA9YupZggfk7XIGcd3lEwJZeJZAahuKQkspt891Kgv38tIM

BvrKTC+xPXM4rn68BRtmy7gix49hPMNVJ+cSx7NsfdTAnd1/XWju4u/rcQFvXasfbXSni1DOhHNJ
1cavR249UZyMu4txsKYyi3S+N/iNP8TTmeG6/pf4Zz/AwfYZdHyGdvzn4FRTQHTGzGRdyDN1ZJ+m
kDl1fvJv/unQIE/fQvWX7PAUpYaWR65levtpNGq5wVdl+CDDyTp/FxvcMqlUD+IFoFdN7ef/Tnp9
br1TR/neoPfvX9PC9zDZkPScaoBFQKKlkl1FyipTq4cWi7nMpyuZw9dlAQIjqnrBqd/kuhtliELlY
fb2j6Tfa9czavQbsdTX0xxhpQ5vTYMYxLJeHQReB6RMTayvzeOSHRDs1DZSP7wCtVd1wnLDImJlB
ckqFi5WLJpgVRNHeVDLUk5HdnENQIulbsOr5VlYwuyrCBwLODy26LF2GrR/NS90X73hDbgGMwDtv
gSdIxnfyulnp3dxHYc630blW+KpNtBn6zks6G3ZufxWpeDtsNYuVqHwepFKce06YCVtq6ralu9/K
M/Sl/HUR3AoZ0Y9Q/pFLFgTfAcT/Px/qyK/BwB/FQA6Mhg6/wyBKHvdq5/6e39a3vF1f20L118Tv
m8rTbTz6f2G44d+kWXmGhRXUxFALNHQsj/HaGkqY7qGhchSrUrKtGbWlFW0wO+gtUYqccQxh32NB
mugiD3ioMkrajZNZ6rgzwJ36glHcQbgOHUkeNNR42yS/znuoLPxufrA7VZT2m08eqDdZV0UUVmbt
rhNa11NKWvazJgK/Vc9BQaCrHoRqGhtseDw5qTVWyyNmj4/0YL6LM42Cvy1vhMDQZYbetHMTqCqD
GitrYJjn0mlyLS5ndVbKsBNos+Rlgaj1lndZwsLuYTPORvf0hY4v7Wr7XI+3AfO/5mMz1RFoj88l
NXyVITkDkQGh3ndf1na3T3seq202wvaGlFLorEbeHuUhQSFzppFbj7uggYV8eqLcRl/BMC8xa0lk
XXTbJicf22Yr8DaPw+gzkg7NkBVL+BTAfXpr/80tajGg1XwjceJ3zRgZaBFTzjsYeUaeSS1NDEj
euwXQZWhpTPCwgiU86Pj6XTHUiAGzdaTjRtMgaoKqvOwFLlvAmHgZ21lARqDEsRR5EE7R6otaen6
ObmOydAGIzibBkNBwF4lDdZUykwKxQY63VUS4nSysPSMv1TixJaWv1j6Y0GENPlxFmj2alxQ2VW
asYMkKslhT74XR1zmo9076jn88SSgJdhcYtFibEEjY6YrVaSlTZCxyo8qBX54CpNJEmao0zlhbcJ
2eQqcB54DDE6ww1IMX0B9+rM1/LfVmoT8Vg4QRF0JtFPbAL17acEjj3pYdr1j1M2UzFoMJG6cocI
G92j3SgHlWHTDNpvSySghvnpHu845Z1X5e5odus/snsVwsA2v6z2u/1BzVQLU8/B5ZtncdC+HLF
SlwbVEvgArQ8iIWGvW9CCYZF/fIRfR9A9Q3mnV986RuGHENArDWaes3d+LVclGyn9JrDnEzTvH9r
eGjuSqqzWFRtJSNkGr4ddohA0WMVpHe1FNQ24sSwPV8ihv/+gOtJA45JqVjqm9mnXbWy10RhBUm
6iwSrPgdu0Y06DYsc47LUFNCZIM1nAVxeJpI3ctlKZ9POwPh/WkJbU3PnJ9Z+U1Fa7qaiWNQPl9
EH54HN+GOB8OrCajoRhUwM7lFgR09ykGTDFfyzsS1M9N80TA4yUDbUBnrDKNUKe4dhYJ2xSsrR0t
TTSTNiQ34eVDdVqz05v47UL+KZt1QsuVguZTk8kqWjd3MJGQ7r9Piy7I/FqhuvI6fCnG9+3rRVTB
VwHvzG4C0qfGMwQW845jOW8SJMnwvbwK7Dg3QDbmIH6rgDUMY79Pq7Ju6QWP/8o9ZoqLdFWoFYXQ
H1ti9wFQH8KVvUi5bYNMP3aY1VDq1WRjcxGOE3Wk9SkEK5KiSRq6fz2roKGIt4BLGynPdSxjExrT
E/va08lhbPj+2gt2Cqnc2c26Q4H1sTSKWYftqKZm44aWiZOoWmuMmGENKutKntWax0jQxcBgZ4pc
5m4p+Kkko3NFMdmKWKk2r+MZ+nN6JgbvaZMo8qkvkZhaqCqHqAFCLn1kKm9nR1p9XfFDdtKo3O2y
VFZMJrreTBlxdwxj/Hc+nr4qT26rqvjMQzXlFpfrQf2qI7YrvWEGVyyChWSqlZTZSo+FoJqz4Yfe
6w2ioUWdSugr504nlkrWLR0DOa811oGG+fwqhWodBiv1cKbQciWv6umFGe4NhOsJg2jMuEqB7ej1
fLzglb7JaA9xtfyfhyXUPD6mZVlNV9nOhBXRtSucDobQD0pXd+3V+JXntPfBz1/YT3zRvJur5pw
EoptbFVT6LR7XtZvDyXW/tuEaYJCSzBVgt3GwJ5UXEum0jjkW0qaK/6QwpK53bpgEdU+q0jMWCi
aCxmQguGytVaDqDqzetMQM5Hil4PpvJ87fu4Yr7OieHNJE6u7v9rjQ9WzWSONApMFks4GihfJaCm
rIIn7pFVPtXtfBcXAZ5VEvNAuM1tSCiLqSoazLKqi6waLZDCVqPKcXmyEgOSo7iqUalx+OpTFcZH
vAn4X+ftlB4fQGfeajizd1GT9qhI/vDlhwOutLrgSJvNJeUxhulLnDqLzIgxYf9r4mFaSbl14lwVE
z2vpzlrYt+sqW0GV2Q1WsiqAYs8t5SOU2fYtjK4EDNJvXNcS7wW7sT43qn0hpcFFwMLQZuiTD3Dw
upjWsvKstZes/sxnPKTeV7832AyvSLNQHFmPNJibGPFUMDGoo7wXUY0/vETbK+N+hKYb2TAvnkCT
I1h5fWPuuvWs/eyZPmyaTmzslx3QRNddnwwwGQAmCvOLQM5QdjS+lQq4DMwo4OIKd4NKvjavCrVI
Uhx9d9MgbCURoGaXLfKycj+dkXbsv985kss7r3lGqDaTLsuTcn61z6oNt7vb7ch9VpOJX02h1pe
Jb00Gk0HFV1COimMtBkMuq81uJ6y6S2kl3rL5Eos+MnzdQia5EOzsmExcT5qnWPAMrZfCnnZyOZJ
iWIZUtut8/1ldFU7ty/q1bWdkbS6gzk/nImPhaC507Gcrk/kpv7Pb15+Uha7bAu0ZNqow1V9+Wad
bpXVfvDoCdxSqkbWE23peqhVLHIk5m76obQSoewQ/mqI8sGkmjc6gtdfStvRyWPTjQNr205Mzb5
KrfaCam46/xWbZRFZCME6uWac+9Rvd9+KJuFUYRmo53VOT5K9T1xvFRZ7jcBJNMpcCf5dubawGy
wKj6X0jXZWP0tXiJozT7mJyauMGmdivSYuRamQcdcHxF6SnOWjpleOblmtZrmOt/KG1JKAJ5H47k
MduXAP162g+ldFd1gGw1/P4fCQfe+mu3F1VoiWcSon0h0uAPoWlFfigZ6DXZ5NzSWD5+8mjVfnn0
GLWHdN/TtOs7rKhZiUkdNe/q9CZjt2wVYS6MGQ6RT3hqu5zpTWl8gL/TT94Dd94PAKIDE+S0pIf+
+iH5Wy5lNFkBjB54nd2dTrd+ehExmensTxrtJCJNuey3ily0Dqzh/RCiadMMBJeWyWp1Yyelp10
jq0O45qTWAAd02HAVw7lFAIKHAXrFQuAyPFdHGCiYEtQZmfPw91oa3eM424KMLhJpQO9wCb8Nj6pc
ECb9+HKdOGf+thHHy7HQWkUTwDI74Jb87DtnfBBSu4Cm9OK079qH1ULgO/99L+UYyzO3/GbPuH9c
fB5XyLHP/PBT/gc/87nt2B9nbcP/50/LGcf+Xe/6DF46wekjFYl6JYTL0aXsjThCt3ygtDKgGR2j
KTRqQJAI7xM4UwnAB7809OX+XymAR5aKlbjC5Mv92Qt1y9WqvJV3W9YQS/dDjdyHN3T+R+ZMgxL
NdaA0auvPld75VLV6+pn8wpaqMptY/6YIiW3L77CqypL11YWRUYZFoJZeC1pYRuogLhRmXURV7UI
9bmAaPOv2rS0d6kJURre9M83USINymc+Rx/t2SN/X3k1iuyd+NtMgUlH+4N5Nbdc4BpJe9819Ogw
g25ev0h6fYHcVvLF+FnLgVLMmXxaywut8XK19IMax1t6Zj1PBubfrvOvMk0QJCKKx06RfKm/rsTy
v37r2AxGKivW3D+Da5hGHZlvTyHi3IAKrzQjidaWFJsxgZsir1xCkE6Y/GK6/ide5YuVJaZLEaNJ
F176XLuJmtpuJqOutF8dN1vDL4U7mu7ypNFpRUhFQ+kvPBN7wOwnwx4b2TAvgnKDY3NjWzIx77vr
zE8kThh+5bX7v4sNX46vWHVDAKtJmKaRUXJcBNSwoSDqFgiuFonWmL4ajlUoSkdvr7hufArnbqMk

K0eRSrL+UI+9vWl8FGP5bkXXpHfDCYt3IS/82t+n7z9bW+XPJqBZjvyyNveJbdHEzG9lvzG3/o1o
IigjM2hfPiFlznZDuC3YAFFVlEuk9VS0sLQelz6xDRbZX4hVlppPW7ttHIWAlQPZlLFxAbutgfgm
lqnO40zuTcaylve8QF5y+NP4TxMufHo22W9ShTsT3/+b5RL167K7l5fVuzr/2v58q/43dLpBLLCQ
kVX4NLBvdm/fF2aoMq0pjoOcJOPh3NnqUXFwg+KuvFactExcNxUYAycpvrirHRyWIwSJ7Dec7EbX
ZU9LUDxz8YnmnM26ZdXdfmkG7Q0yGds2mlgwmFM8yBo7WRh2Bc37GgqSjNZHHw9udcyDPPLAFQGm
E6lLt1+UPDwjJdgddKbG6xq5UpNR3BOqsD/EaobgB9anjtp7j30UViEE8v1clbqtlpNzVFSTYDC1
AxHLOFzThdU4hetBaYwWFGkWjgA7lXlPjng2Nx0vrBAgeV3vHn3Ll/R6qcWLF3MFTa2mp19IZrt
3q6r3t3npXdjg/fmJeTu7fw945cfefhRGR7ekOeff6meugYOy/LAM1YbFZVKouasRGJdcpVtvkV3M
lWirC2b1BTacpsKlOIR4Ggps2Umz334ozLcvy7Dgz1YtUCGg3258+LH5KmnHpYbV/rysZ/7UTk/u
S+Hlx+S45efl+d+6ofkUq8j73rrTfn8tz8qj968KoPesAYc6KoWRKhYuVFbV0bFnVovmXNgLSocq
pIenm+EpMlp/fyFjhWLNIDIG8jy9duzhHZ3dkzGYSVYyql2oH8vJ7FyQqiI5xPdQg14whltlNQMSLF
rzw/XStNitgIcWC1gZV/iJcjL3XAlXTNJ8kkHSxfapq0i/13i1YP9ON+rPwAQlOYW0rwapld6Xc+
o//VxX0LlieguDPXb65hbfIClsjzAlX3Y8mdYbhqRXJYI/qPswPva9N3rUnL3ylgxvhwu+GrdakFu
DZi2aDRTfx7Hzf5lcOBDJq+5hI/+P3/FFTvwuwcXgPNs9UK/bpnnpHL1/b15Y9+SK5ee6t+Rq+3o
z4mBxxfjMpgSmcJS0TB8fxCIA1+7X8W5OTAKqlV/jUHixvY4AjGeCmG38bTplw5eFhTSqvZVIngj
7ztvbowcSE5PDyU8emxUtdbH/k5+fff94/kJqzsFz7zbn6sevy5KOPyP7eobTaA/j0jlpWlYzaS
KVe5H3pKugUdgC63RlKujxTK9lq91T5gmWQ7HEt87pjKYylpY+u/cMZh331ZDEby/kZfOfJQs7Oz
mS1XNfSNFstMINGEq0H8PubeWmYwBRShznCd8VqrenpHfjct/uFEfwRg5IvBp0HMT4Ti/pGKzX8H
I8GG6+WpnzcNj9+VpLl2Q6++mG2HntO2NPxh5xJyim9VHRIcAcvNKiU1IGSs1I93hL0WL/5qm4ba
/rBg3GNOQceEyNmBcD1lOreUJJRqtMbu/uyC8896z8+x/4F9Ld3RW/2ZTlZCyz8bGY8KauHF2R+
PxY7tBvbYVYBALNh7bbhyrrSeudJ3WZIMHKiLOW/m3GGmrD+wa4Gk5hA6gTaK0v9ZamsNRj+Jslr
PDi/ESmd16UE3w+LVmERerkpI4GczRiv9sWZ3hJ3vOe90jbt+Xo6EA6WJQ4tJ21zIxak2YbnOPis
JdtbWb4bLSOTVh3Ti1oYNERR5a0ikWdj1Y5VSobWpWEzb42n2cp/HaDBRKuSqAyF8t0EJZFyeAnU
yolWq2kD8abgsAnitNpiAA9ePEHyRVOFFWslMsQLUvnFeuLkfnveH+we/P1zxR42M4wOmnC8JfS
wv65kzdXaIUbXzWi79Fs/vNdH76VGL6Dy9Obzut/beAap3oWArLriuIElYVYbXmsCfRrptUrarpm
JKu6tw6VR5c36mjm/qQuksMm7AoLH6wzFCMKJHdFqjZXgOW9GHQu6n2clpVLIPQl7f//+y9B7Rk5
lUmuk+uUznd3LlbLbWktmVbtixsbJID9ozNYAbDM7PG5sEzPDBhHm/AngcGFmEeMCzSEJ6HGWYY
AwM4EV2toJlyZKVU6tz9823cjinTnr72/9fdVvGHmyjUbDukWrdvvdWnTp17r//nb79fYsnqVopU
b0xJ6Exeo+Y8EkOLkofViJ6iE4BcG+qdpEMgstmlMrYmyldiVgVmBCmA3QgSlck/877K7SPQ+LKg
avEIBE9KDDEROHux902RdceE2Nz8yVBF/kcslarVRGjdvl9XczPGgrbDPpScA/LNBCq4pJukAj9I
G9E2Op4AFPweXiFtRGKs3fPg1ycP0uA+wg5zCQSNv9hb0tek3LemuMUAZ67zblrFmUymhgAQcav6
e5sC/yx1lzi63SEsWLU2+QNKPBQFEShbtBrU6e9KUEjZnXJKby5mjd+E3EHG6DJxpp+oZ712WawX
5bGatj2kxysWvHTVBZjb0ml2DjU3L7weAFV0vFwh50SVKUcgbvl+y1hgTBdler3eJFBhxQzngbv2
JT2Zbd3+H3y7L2kuAMWQoypszmk8MIIX4GG9x3RS12eq9Bio8TGURKGOopAYKAAH3GHUBYoIpgBM
QAQg12BF/sIkDvgcBOFdFKFLJWPo7iEjoQo3mGIVhOwqUmcRDYOhNTCJMjXCoMrWjF/jTkPBLIpI
pvf0yh7lM6Xyfy592aPZ/k5AWA4gu21tGI7zeCUsklFlqZysWUIQEL1NBTCNCypNOyLEXqgfDESv
kcFGkwkZpEK8/b2JuUKJQq629RjI6409onBsnXRqLsj+OB6hQ2215GpHPwen2XYWeXoYiRwzkYuL
/fEL1UBcDmThqPr02hSQXGrXKlTfzSUKaB+d8duNg5+VTzuPcQXMHpOtx6/7IvB6RU2m8JzulYax
XNB2GqMhx0b7HwYKAcA38mZIp6MiBIV4LEA7DXtB5sh5iy7rU2Vowq9ykiKTHJ+gCVSZZSSgDxWKU
lE0dyVsFZ1Unky7VFDMEokv3keaoJkleRzYFYSVGx8fSq6dyMQPvH3Oq0g7iaYqddn0z5fNktSqi
KKkMuB9wY6YTgfuOT+EopyJiV5JgS1Nmp6pQpRUc/MK30BxaA+qGduSyq7KQ7XSHXt7TB2BhiXmU
FciCgn/HYEEsr8X87AwctCPEm9Gvk99kIdneWGKBHwRAwGV+f20s3FWVPgwUDAE9aTIN0ELpzovq
vI9ft6ouyVgilrzsAiBjXvb10e/AfLbK0guFZorHcv1zpfqc9C2ubrb2uKNELO3rkA/ObzmlId2v
lCv+oWEyM9ETvv16Vv/z58ERaAoGNrsFQ+yHdw06PYKHodcAeeriocHzBAJIfqesNeSCrAojaeCy
0WKCI8KBXajVagdAScIrxJYkhJZ/FNVN4RiMV+MbajwGsPjQpdiguZUFake/zdW4mYyMwoqBIDvI
ZYcjqU9BHRVPsc5o2PthvN4LakBckD+xPNlirgbhoxwOhECmkQ8NUbcTNHQYQPNFzSpmxqNQ1EMx
gXjFGZG+WCuHulTX0UAGjmxoaUjcQmpKeeTvogewAcwRKqzqBLzPQKJucteEPIeE9GHXHXOvkiXz
z0qc7LALepDJklc3gxVRR5q7jBQEK/10YcO0bZiD4zNCKbocWgsdlv4pQjV+xOckgSWWzrr8PXni
0XpTY8RKQizNhhvXJGHPuLR3t9Vu1Tp/BABSbWl6fcmQulh4IOdZFqIRKh5T9IyiBW+CtjQRii
LYELGIAYADUD4vUlsyDYEjDPCcWpW0o9jyw4CNMTWuqRk3rwIv5rsdGyiEozmFqMjNCdaZbQE+qx
ZZtWfSjGB54nCxe3kVemD76wI7C0oqXnPV1lYcVjyduKpoHV8++ijgWe0KbPbcnAH9LyLgBZIBnR
OvFY6+K/itaH5gvRZ4HCKDMtpqeyosxzofoUo3o98TnwP8UsIJJdcDTPNAaF/AiAiNWNr/wPGEa
nHEHhTgE6Ca5vcdp3Q0UCJbQnDOoT88Nf8tMOUDpTuT70HeKyrRLTcnYBVcm3jLKT7byctXCMMmg
/aJaLzJYXMzm1852EW+DaPmDRgk4MVpgAwj/ULy0WebUZtfrgY6zU9FFzSaCOM720An4TidjEdgw
7vFL8+FyL2K9UUqVGpisOEK4hBYcSshQrVB/5mlws4PAwIYQpgM2csAojdt30tYucJQiaOtKRGaI
UYLgWNLS2Wg6YOfkaVG4GAEEb8h4kZ7CERj/F5Fzt1A+wK2f1SqhYhMyA8UZQul6vPFxq4+E7ifD
E2mBi+OCEA8v5GqsJw9NHJpjp6pIRVTwAXCI2OaStlO5F/ZAPieyb8hwYGSVYopm0RTp3oCNZDNI

1EhPB4IPywbCnc6vMHG4HvCyh9MoCTv06izrarbiSJec4tF2QSRX4P+BQYLInCE9kgdMCyBnrNAF
F2FH0b1XpgeDfTRMTWVFVPLiQu1/UhXLsVRUHFLFUwE2cP2ep3/QieResMA8TC+iJX0uR57xvoUH
ZBsDERgaiTsBPCMsUgMjMjC89e8cuz5pQ/kStVWw3QF4q1EihKp+mKZIRSTCjAa8o5MbIpuSw6D5
QDuQxoRxxgJqThNeV4eThikGidwTYaKTKeIzIHgc9B5h0ML8oMm+NRNimgXilfhNxZCR18GboMjj2
sAq5xRiiFTICeU4sprWqWQsaksq3qamfZ1GE2qeNtKe0hFqToDtbfFUPib7s5ReDjwVDBD5L24Kb
wJS+YWXtTMZ0yqqzvBsM0ZGS8uAiOqe6VAHKVLa+q2juY6nkQjMtw7aydpWrzEA26APWHAp+Uc
6UKVAKdW3hjCfCZOlqAqhly+HDUESK7pN8TmhlDnpfQaOeMDV7n7uaF/d2dLeqsnpylnkWB7RRrr
+CTeNM18qIv1Gsaxud+7BnrUwVeyv7BgpUZUF2O2Xj449nBV791s1Rt3GG4pUm/3aZOa4d63TYbg
S2D2GAqGA06KtQT5v1EVT7Z8LEg0ee0Dcy1RkJfLulqqRKWUgaL3BRe0JE8zpxw9pvsDYRmeHqdg
KwCvYPwGV7RUZM88MaCj7VV+Hfln0w+I2apNbuh/Fs2nFTmW5UBKa8+LSQBeK/C81jaTUKRSrGA8
9U5VEXbAOKCaHhBDJemeljekPcSmhShN7U1+bgr4TJYErOoK2gqU79GKQ8ojihFysYm7teo017jD
5+n1s4Fqi0clTlHDB04fa71dM3SiLFAvyiGC+9aLM9J3xi9WGCKIxS0spkQcYgZHwn0Gao78ju
GLX8Sjs72CQ8BCH13P3/d4PflGW9qK3/5wmCn/yY89Ynypj/RyPf/Ac24tsI/meE63ldFDGHO7Hwf
4F+q6sgcbxj5/ySTK6QpbxhjH5hviJFGSzlPtBSyIXRnbdVCDxVBDDnleeiD4mwUnK/bKIWseRda
gMQYEMUi/E4Iipq6T6uYuxXxj8N7RPx/BJ+T0PfNNWfTUTlmMgtM/PJScEVGxi8PPihgMBS1zXZV
QSQ7C9VFV4bkYKMnCVvhs+EnBZkbCbNKFBR5U5FIsQXOKG8P9pKpAbkRbEOUQvasRjN42CitXOZ8
+cqe9hTlK80ZhVtR/JbtVkB9gkfDoMFyMLnDRQPEp1Z4J8tSsc9EdOy8yXKcRojg/ehIqerLB4U1
fbG/FKQDFvVSXe74rn+F21pImL1WY89Y306DXrUy4w0fshlnMex0MJA0WqimlgUkmVwW8JEW0R+H
fGuprAfKhkMnwJp82SCGJoRjgEPq0CoZE49oKWY4E3NGqhgeakiNUOYKfqsJMz8KgpQeSM8g4R4U
9LCRALbZUmiq9eJglHKQo+nu5B48ytBicJ8qBFGwkU87cnairUQXhdGBuMVzwQeJ9OYFdISznWNa
VFLRvIUOEL+bds691WE3+JphYw8FYOGt1fQSEPaRb32Onu7Al+KR6uXTrODbdCAfwYjRa6uRg8dC
Y0xdijskXw9frHO7xhRqb4kXtT3KyrvBZRSpqSQ7+Zk88M9LlUq1PFGGo934v1DJz7sVZp/VF46c
ikOR1/0XOVJt/20kuG84rFnRE/j8djf/AZHn8PznkOfLFbqbKSOMhZxLYH6FX2L5mo1qhZyVCmVK
ZdzBeSfJqoPC2QQpkN6Q632DfZEW9EppbObL6JqKr3URBaieiJvg16oGKSR6YhAeSQBV0BQSjiLU
zFy5HvKUFW7Bbmtai9DwmhpxD2w3i2gaQzQEgyA0rMWjAShifyemH710LXFcWyLNHhr14kspso8
IeEy5Lou2LECI+B/pLPnCopEtXzVZ4V0pJhZggyLIpVoQ+TRsMgorW1VTa2vND1KGX3SCQ+sLEkb
Gy5UpMUjJGE2A4zxvlyTb5iGseC4DXoZKSvm0uFtiZoU3312ijsbce15aMD7AK2a+9MgkHv+n/9c
1/SELpS/tt97Bnr01owNrIkjtscCn9wfq55V2N+mcqVKpVKJSOuPZpvVGh5sUILjSitzc+xt/Vlv
ZoO2ADVTBw52NrWlnhkLMipJzKmHL7wKsrc1M3OVD5pai4jAVZo4aTpgkxnglaZCjE1GCFJwd000
sihTDWT9WsvG4Sa+ElBM5ooQzOzJy8y8bCkNF/k/U1RUJ01ptV7eqoVY9g61NZs9lMRrG1Irb2qN
GEZ9oi211EUO4/1+0yjqAzNnAEIJOZaoX4AT4zWfj3mS5fPUTgOBjME9oyEzVL5LVAwbvHPCZLFM
y2SoRVUQC9Lvs0hD0X9bfn8pm0HFmCQpSow3I5fKsbF5tL9o9aFu8Lh4IxfqCRf6nK59q0/vte6e
YYtlu0t/YxN4X8olWvbeTbIUj5HtXKR11f20/JclRarOVpcqF0dwyrbtQQg0e93BdwP5od2L5BKs
3gzHYoKSEDU3ZSZyiIjZcjCeJhqDVcYjEheqLww0z1WoqlOFruVo4wChomrzOVEQLOF6B+QP3iS
BWHklQD+xORs5AHUFZCqgb0Uq1pb1Z7UgMYWoTEbn6X2V9LOqr+jwJjiKfWue+sZSW6s5YOy7U8i
MTsKqdFu0vulXgibStbLkqLaoRh4S+ehmKmt+wL60bS/eZefZLyMzpgleoU9DfEFAFBJdRRffK8
3x5vJmh2ARK1wwF43U21gUadS7F9ZUTn3Zs9w/zc0f+Np1EW4f/2Q/+k6hdbXRPx76w/fuGevTX
Ica8wLbZk/51waFv5jLV0Psyj5aJiaIyUyhWSnyQ81kqkmWIS80AxSak5i2Oi02XtlyEdYEVw+CK
6kKEs+nw01BG7mqv4ogMUT15TrVvUulGSqhMryTYP/0cLneCCDHI7y/tuY/ShWvk/JgkgnrDUPzC
gO7zDkcr3rFmZQqtbcpxWxmGrOQV8AWAB1I3uypPBXXS3pTGfeVSh2erzV75LqkVaMkKEWKMLM5L
zYvKYjx9bV5Y2uNxtTijS5MbWqPehQmSsEAPdXxGFje1AJ+jjNACDZqKLMjj8Xz+GuxeZAI5HXIY
yXtQN84L6AOaOay3wbXqtWx1/ED/v2n0iS+jy/g4nXf91PxP3XBXPetPyF94elJz1j/1/RyZqNkw
qBwxUNFXNYgCUd/mqbGo46phrjBcACepEQEPwLpd6YyHG2IAQ16W9L7hN7NRmtD1XjESKwZHaJS6
1U9UVN4kYBKimdVWRnYFsV0R0i8p3qtMALxLLww+SJE2HhaHQZlKWZCDeE8UsatZlJ5YVvkNFULY
vG4RhIqXiS0UwAPnOXTlgqxAW5A2wWFLFO3aBABWEq3ViliqdaOwQYk3lZyc1NFCoa9qxErsh2mP
EdVgdVGAmz02c0WPb7Roc3BkLZftgmFLZKCFcITzMeGw57QwA6hgM7vgeEC5PTYgGSD4Xw339wvG
rpZOFZcUIa659DIQaiNIh361abrx2kyMeMk2owBIN6KjpPf91MiBSpyoHvG+hTa6NQosXCmxvq52
zo503b741H/d1c3t8PU81ORREB1kw0BJGfDIJDCQAPgJ8VbiClAndxY00NuOvwlUVzVm9S3ZII2
k6ZXLHFIew9qC4ACaEgUcwJumAhhRZCCBiNozK1An5fMWT0NyG9IZ5VDX5LZrBfMUefB+FxZkjuS
thsMHqGKjKfI+MQEuG0KjDxNcEibSWepfQ/tsr9DM3kIOhFRd1Cup8rFWgtfiUFKVMx90/dfvRhk
TsrfR61GYb8/ZnVNeqHIRU5chkOQhrzJYDdcDjQcVABEYsCmhDlex2RIULJK+TZrqjwye6XBJQR1
cnycpRqhFUEKQ+nRJMB57eFxmACQAs/fzIelnmDdaEm8FQeCfLvaM9Yn+oi0ufts75sBLHYel02
p3ffeTxRx85d+HSCms7PhutsB0CKQS8MHYXKLIWtIYx0gb/Npmq0MTOKPiYJYvCv7XNsXbiQFo7
VTtyXUhSg0DiNr4FX1OzJtiwQN9BXfiYX3BZSyjwqPbEnImpM8k3Ke0KhK7jmtEOOzx5mE0/DoK
Z8riyJpbcisK+kQWlOKqpZROqsiS3g8XR6CzJm4f3uqo1V1AK1dxgmvLuttGPj2aYTU3c8pvYAA
+lsyIYie0NPOeXnYeIpkUpxrGQvM3VNqAds2/IQIWpI5EQkzuGGpLIFcoy+4rCHtpVMeelguE2r
B0Mpg8Gvdy5R++9Jo7jw2maOE/lrrrx035RPtenfuv79oz1qbNVY7cV8T97mmklMKhKpZS61vcJD

svG3f4wG8Ng2bPAYKd4YZR2wCpoCeeQI2t1qzegbr8nJpBg8aDKC/U1LEyyrjDSTIW8WjRZMLzoD
zqKPDuz9GKNffs+SeHJ5NzZiY9X4a9AC9kyh0q6RyqiysAvg1kQhuLYWrDVRQYwCuivIkxGeIjBd
lMKYvreJPGTWRLi9fWYm8ptDa0Wrnq5s7aFeFhrN81Q6AGFKdYCzKLnw79b3eqKrAbuHobe0R7DZ
paCRxkjfXx9ne62tHZQdHJsJesB0vWdzYtskJb8zHXLBAoXOa+hBKSBRQZ6SuhhnQLmXZuG54/Hs
WkPY2PpzOMPfRXf6Obtv/49TyNs6KZ3/oo8ns7jeULybfXpVkfSRfO8bPbVatXS0SMHh8PR0BsHo
YEFaA0J4V3BXAggup2T0BrwQc/IC3k2hrAvb65Ssz4nFC/sqJWqG9Fs2kdEsLTnMAyt1G6p3FTgd
TC8SEa3xYiNKCEHhS3xpnmyOXyYVe4nxSYwZGhgQR8wlKRVXkdmASlBQMvpwtAGCXLQAeDqRPm
6LNgp/ZClssVWwoypkKA22YyuAMreRuGEoIS77B0LtIW8a62GSq0i/CYFSpMTKHYThDRRC4CxvtN
uekhkzP9DnsLeRrcjqkkoBlgjldWme+Jsyr+qU5DhZKHDApQdfstsbE+QQf2H6VaY47fljcyjmik+
OUU2dDZa198SLiHhS7GzYWcLmz3ts8cD6003mpdvInJzFft27f8Z/f9/o8GQXddNjKbz+9XGpIe4
W907Te9+1m/iu3ng6HK0suMJ3mQ2fwEyrqGWeI/GIR3g+Zc44HJ2iTsdjrU4YdjW/q170E49Br10
6L0zWEzh3TxDaxwfnWDRr8qohhtmkiZ1MMoU71w4W1AXD3rqWqDJWMGcsd0DbTV7MiW2Vn0FN0cq
rEcVPIiN3IKyAJGIdImE89XSRTMmjvmHoPSvX7GFqNXQyLN5EY3tm1FUAfHgreMAYRXYkKJqWSn
an6LgpK0wpvGitgfuooQWc58WTX00rBR1WmEX6rsNag0xdWpeeazxdEpNq3xwLNFExxPBKuKcEim
6plB08PwH+1foD65x+ikDeuxx67j05c/yKqNxb4I7B3HWN2dy5e6151PqdNd7cKlApyGWWawwG2
5wr53Lsly9evHza9xc8bFaGNxbcnY0tHhEcmr8jU78ix96zqzkL3tj3aVesmiXRM24on+TwYVe5
p+tob7juU6uXqk8cf7M6footuwk7MsfNceLe5Iqfl4QcoVBm7x8icI0oBx7wNwDV5UmqZU43YhS
iX5VqY2DJqhmrJrtqKkkMQSVZFTiXBB9Eie4I9QtHVNADmkWKsFmTAULIaRm+Sv7VhR6ObxF0UiG2
FHkkcJQqr2rIVy9Fn5fqEoLBKRvaWIJCgtGhQKXGIuEs47ymKQ95gzGmKkqsKY6nZG0SQacacBFJ
HzG8lqA9znM5bSZNre3pXKOW+X7iNncYW/InyegWrnKIEyIfL+gQnPkqPxyYlKHwx2h0sH9wrjb+
dNn5Hke3+8kGQlXOSIEeGQYKsbpUt5tjCSx5+YO0r0P3m3n/LnGJDRefvdddz/8optuWkv9dN0b7
mQA+id8bz75a9+pSOhiZbz15kF64bf92LnyLT9/+qyGqS3X+Ox4GH8zNBC7V33d2/sWW0+xmLsjT
iYblmm1xmOM2QXSWnDFi3GQhwIQh8TA9lqSRzq8YfJa21xTPVUB6cczXiQshkTv5JnuaUrPVV8LZ
kNlrE7CuiWMhQcOEGFDF3X7/CMUDBESJkoSQ96fQlRoATPounJOHbDX2j71EO08cj+NtzeKvSFAC
higDJhzbpzz1dwsqTE4AWJo+KBibdSgDgD5sytbX4kyVHjNWH+ldLfwPMnchNTNzGY5eo+vbTAAS
2OVwmFbV7xNUQwARDLg+2QXazLOiJQCD2HMMAXhh0AKgAELjNRhU+m1W1IpdgsluX+RoKEsgYEKW
YBqz+UaCytrc/MH2OA79mA83Me/fc19d9/zkm5o+9jYEGp/HoLwZ+0Sfh6BIRLPfyRrWpaUnvjgb
8N8hq5t7xw/evQPHTOxhF0vVGRfWap7nuzdczrjx7xQfA6NZbFz+La+vTnrN6qyaqKhgYYYY5YpB
TbV0ldWQtXGUTOhUtjRUEUB2HN+HPa2KRix5yg0aNBalUKKPIKhUHiiTjvNeFNJRmM2RM12YgdC
rfWhVfyWCB4vixTKShGoTBlerlZ1VWKRMady5ieJkWXGxPPNppZZX1qtKZCUYlhEmlARDoTDDf01
BCM6/zaKhVKRY4+fIowJB7zNWNwn6/DyxXY0Y9oPBjyZmYp6tTZKFGiSM75miCvgXx6wtFEp98Tn
UHBzppuxVmQ7wnjjwSAYWLwnP260WjM01xzGclCMZ4k1w3Hyeseve/eq7bHqPuh+OvPRRC+FwY//
RWl7MmGmuq96UmeNROQg+rxZBJCsltkQ6kfPnIk12bG3asb269ib8spnkuNalWptWGWlavs5vqqF
DuoUGTf5dNmpyv9UWkPpirPtYwpAEIYdlWOaShpSFKTSkrNI4j5VF4/NRBHeJsKusBlklDzlt5iN
OVrYu9qZ2oiRvqvjaaihGLUyWJvZbKBCTAentBQly0hLobkASPUBY+pF5y2aQwz0UsDobGlveoVt
1M2IUXRqnpPwmdSNws1gMEWBfnlrUlanm9Q005SjCiADTSeRMJgOOb8v1ht0JCNEd65Pwxoro6KL
nvUXEk2AdzFWnWRAG7xhxw57KyeoWNXH+dz7mej7UtrDM8CybdV8tmwu14uX0wMJzdeaNYGjmmXL
d59L1xcm49SesV2f3j69KOP7CSHltYajWaC6rJseNiDeSOJg/GesT7jtIU5lqOLv5/tZUXLQWWwa
dJhj5B3ff8FR44d7fg5f2t1fXUhTTNT2gquKbOujpMXjG+fvRm8HOQXe1CL63dpsdbg05vqPHlKw
m2q/JOUjIPC0Boy35qAlIwidVWocmJcTfeILT07maZqvA2eADkr2PV5iQoiysxyZbCK0gYSwu0sP
wPdYwTPmEIMUzVrK2GqSHMoahJB6eO+8Pum043GFPVphTHW0cFui8aQzyNGK4+JgjeSYtTA3omKM
QLwIRsDqsCYtMnn8yoS4R0M/VXglfHVtHwJywdspIt2kz0vPxc0r+yNAXUyeKOMRgNpi1k2RzqBS
T300YscyoIZEsAIjCaGgSKzG/VaVtnyaqaVtYuuUTx65Ei70+qWur3RMD6AX7++tXN2cXH+o2ykf
colRL71SoLwvTD4GY95p9VgU0+R6MWXqUoqPIXydtmFJIn/gSPKWzjkXV9cXrq32Zw7zXlQgkJP
VighblfOrB/ma675ghVSuxTPUeqnwmHbBfXLus+oAKxT2ddSft25dmm3yeCgEKhrUFNIwHiG5qex
dDVY9sxFbMfWARsA4vYoyUCcLYBFiC81CwF1qObqHo/JEi7QVT1WcKdFBhkw1wRDmq+grI4LSon
B2LE6ObjYKlGj+sIpR0OqgwY+LQI3YAVWi2Cni7Ab9nf8CbGBtCDuTh700RaSAvDaAZy9cWjPszw
AgA+XjfyF6tDyl+hy5qE7HQ855axIOG55L584+xp56wt+3NXGaIYrpqfSmzdgRvrPMBOMqWXQ8Z
9s108HJ649zqGv5ju+eHA66r+912ke2d7btKGwZiFJAeA6whiII3/Osz4LImBfHqVcO33XG3cfPLT
mZorWCnN9JAOW1ZfOM0ST8SBpM72UBv4K28U54VjFbL/e7haPxK2/UcXyZbUt3vCBh6VonpI3V8
4JPvbixTSevAJO/GqJGziUhsCUkPapNk6nI2xSYQKyqu0AZae5fUw+KUwADjYn0hI3tKQ410Ji6+
aJSII9CyT1h0ArdY+sdgT9LFKuIAgaaxbMWi+qt0m74K4MG02IYPKOL13RDSambCkYo9DMCEdLks
0iPWFe+VW6uIolefVK6tHqJdrbWyEUeCO9OkKFMxYmpZsgcRWFCUWYL1erO9irZHJX4jqdg15Mxz
c0v0XDU47vEaUgdGkD8c8+nLkcwrigVDGkMoDEAERLKRgZIA6Jg5KactPPfOOaw3ij4zupNN5xY7
LS7zVEWusmwzbtzfv1cqdLspfx3SyPOiDVB+J5nfRYcIvAr4r6Uu+POBwyZCilHogiOKqymYsn6/
X4UhFE7To1PstF+yLacRw7s3/+LpVLpz3kfn1iSY5I04ReaVar6plCGwsC2d1ZpdXtTVNxxGsZUo
2WiEpplMwoW8V2KDFBVRsH+QNZMvhGhqjklRLMUy4RA9oWH11eCwwBBoh+Ldk4QKpZBy9QV21A9T

MU+KFM8pGdNs0wXsqyZl5xtaTICNmWfiCSszQydpqxqahlWKaKkUzXZbZPDIji5C2TKu8Pj5c1IQA
rtkoTYvSuY4xuO2MP7Ds7b6Ia2ubdI25/u2V5b80ZDwOJR7mMv71FjcR7YZ0vLCCuV9DKZPOOQdC
x46EW5gQ4pjQNoP+jlLKNl5R+H74rCH9tiAy5blliuVemdpeXF89KprcnOLS4fjOJpvd/sW0gd48
XyhyB852POsz3SzFUyFmO741KcfwsrMvfylJzEeR6lWW/Im6LpEUSxTMMhP4Qj5jxzyvXuZgsC6S
/NzD4eTSYk39ldHk0lulMVsgxbVqz1+TY1cDlG7HPJdvHyeDi6t6KlQVYETkHqZu3OfmV700++Vc
ScKlyucwZaO5i2Vq1KmoYCKNxBvFmicgYyMxItaNiB30cXlPl0dikuGRjKpWYSp9lQEa7vtmWRGv
Cb1XfBRAemEn0+nwQCA0KkDDBhGYcgo3G7qL3Vl9sDBJKIzZ86pMTa3QJ2tC0J0FoyGVKutiIBAB
2Tq4E70oC2EIhEbMPDCbLBo9fTbq1SpVYXzuMSGLDdzNBh2yags8mcEmCIUEvZ4kkp0Ijc8A9IsN
qQ84bhZNh6acZhBqsQ3bddqr10sXbiwbnOU8oLl5QMr+cRYG3e3xiv7DsZTgvA9z/pMJrBQAcDI
codgUGyTb2BjJ3xooTRYi4VVC0wligMs3ASNiaTsMI79sJkEsxxrjXmhfX742H3/xn2Ox2M0YWRM
ry8ZlCx4JGfK9GFtYvUC0ZCGDadppkWiaZfU835KyFjSnqIPJL+KYkCQKbxu1N2Ble3izlvzdSMK
0jWUs7dkiHneRwKpgLcT3UPVFeYMY8KD6q5kIS3MFFgfhSf5Kupyc0wApjFMxoapbGY0swazSmz4
pPH/KazrZKno4jGz1/bWKfNrR2q1hYlPwWXMNICFIRwBJybIsqplktULBWE7BteFe2biQwcpJSvL
tI4GMim0Vg5RpVKgVZVWvhcqfAtscXJe0GJHr3YQBTpc4KAei4nTCGnmXCSR5MQf/dcbeFAki8U7
Hy+OGeb5jz/+dlxGJnJoGlogvA9z/pMzqrBaKBrSlsR8Wy30sWNKU0oG4EZx5BvZyocnpb4+6p60
Db/9g5e1acchxNXMz3B2/m3hVHgJQZDA4um3qgKq3+vH9OFC+eoeuIkOyp9fgk9UezRGFPdKXvDL
jAGl8qg9RQ4obRxYCI2hrMNva1VPMUZ53i8wUD5fAqsYM9lujmZDSXMiNqKLtQwdHtIHHRFeFnCa4
qO7grFVvatUVXYZ6fIqAxd+KOCWU8EKyWVkOmYXEJglM66ymVCiYcOWOGB8DrzD3ffdL1BAGCB4j
0F61t5aJdNvCAAC00M1s6rYIKwiRwgTgr122y1yGz4bs0nlWkVC24Bz1jx7vWLjABvuFm+oHbp86
QJlW77Q8CDMJnamEhjCMLPVxVKy8xUqJxEcg5ZDby+Tx/5MC0rVze852CmSYub3DWmdXt5OD+/
YkLeco9Y30mol9DdvApWPuOux7AHy168lOkqJRNjSROEgiKF9kwFnH5lJNYF9EZflzm5+zAKL/5p
/7H6C9+4pu/2/L8ZWeSvCKJogoWth1P2GBrMkN6aWONrj16NRUKJV0dTVWZSNQZwesVS86LXBmMi
QILFLtQWGRp8qTQfFUMigY8UoAxtwk5hXUOKRszwjsYx/Hq5uUDI5KL8Xgm0QvNUbos1q5jwF6
jcUtDDNxnoiRlWbI4SqooSXCv+xqUPlJ/HjyqaSkcL4G7tgiUyx+EstKlGsEJ3hSIWV14YBBY/fH
xVg/G7U2yCTjR7Keb7vcrjbFLig4ZSptXGRQ+EJlfiLHI4WqN/bJL9Yo+E4Ir9iUzTYIc+v0JFjJ
1W4HisahHjZOAvltaZQ00zkvmLjFHOacEBNhv8n//SmeF5NuVcw8q5bs71PGs84vw2aBlpWDe6/
QvZnrE+A4bKoY9S29Pom5tuvD77lKcfTFUoKugigxeQdE6kisnxL9sqG2p2gL9vGNL7oMv8WEU3Q
Ru2rOA3v/f98d//4jvfbNvRbY7jvCxG/qhnMhVNBv2WrTV2qIc5lqmgN/VEHyiih/K27qeYvyX8
DGTILMMVEliikeOxwGZiB+OvWXCxmrfqmAV9l0XG/LqBmGucEmgTxRsFHDk7Mxdnlz2NmSAXUlp
2Eo8IVrU+Kwn8M0DxsLFjome2SYHkCLTBuipXAi8JxCb3qFoU7vr7ZVyROlK6tbVbfc8QkKY/bMk
x5ZlZrM5Le+otpaZyMtcuiL90I118gCqtTrlGtdpoVmhrBml6hczLORNgAVpDEbvm15Ir2R9/NS1
cZGMr94gNobF+TzeRjCdzOpTKN9I1EUtHM59AYM0cLUEiKPGD3dspzHxRgQaudGnM7XS8awH80Iw
veM9elMxEXASS2oRAimjVkh54aTV2WqjyceK4WxarZ4/Ae0+YJpGEdVzEqP8zkuWqYZqXOk9A0//
v7ZzvvaF/Nb8Qf/wzu/wvXy9zgZvQDvF20QnI2qUqnQ5c1lWl05QJ7pSENDMkZDgHnCYQdHyRsG
RTNcX5eaKN2l3YubtDahbNUK9apXKik2jg4IKAGgLYJQB4XMgtB/aqKKslwAvH0I0pEWqxXqXOn
q2stG84vOaETnRuYinqBDQ0OdzkTSW/uI+vr8jPmw7Ik+4Rq01FcQ7rPHhagDLSGZULvk/RtwVkk
Y22zXn/gw88wBtXoIbRER2gIMaGkC/4nNPNKefZki97Hkaif5pEQ7rqyGGan18W0eTJJOFUoq3mh
2NsPNACSi3dEiKXBkYHCFahSEA4LahMwueKXH/icwZB2zovl8mIleQdh0IxfE5cI9570o8t5Akk
3FsFaqpW6yknNfwRm3w+Tp7xvp0Ht1OS8k1xPFu9TWBAnCm7HxXeGEs0oTXjkFzvCavgsHygr2fv
54H7S2mPFLdXvns4zX/128lf/3Tb39xEazvdDzvXmnIrlXkRfpluYWtdrbrtDi/T6F+NBBDATOmN
KF8bezp7FEoMo8oQxerTVrktZ6vzEnYF29e4vA1IYsNw7UyYUmwDTWHakMdYMJhn1fgsLjAz/cVp
y4bqzu3rHijABaAZgyp9o1Ra1BSKFDYvzkZJpK+YKcxolrfL8DgKU+YUb6VYH0gzXKhZV30fLYWCQ
hsG9+e2j/8lhYHKU0v1ftQa7rABJfJtNwVOOSCX7lFzYZk9LW8q7NurzcnURc/RcNCjcRsshyFlu
x3J2UvlulScwdTR2lmlenM/TdCXlhlCzlGzkSxiKM7l/KIQhSecCiD3nIiItCnQRZS085UF3gww8
6ZRQy084Hs9tBw3cXKSx2Zxb5BZtrtnrE/nMRwM6R8OnGdXdnKepC0kYEDTrGWZeYR/wx6StrOxP
paw+0KoBu+LvBZV47/+mX9Fb3jPf33Smd/w7343+fOfEouNwbj3p26ce7Pr5UxoiKJV8/iZJ6heb
/Ki9AjJsJHtQh8F/ICZWAd8Too0DFGB5+SpcfAoh6dzNN7aYENiTwwtGDY29BkB28uGfXL9ApkFD
u843Iu3L7GHKZBdrJDNXt1sLpJdW6D4ifugPkCqHE4WCCjVKC0kFd6rdYiG2tBGASnE0GJsDzE4
sGlqAVQRqqkM+Ta4VFRqBKvmgpiMuvURNb6zCP0xBNPCughzxy3BLuJ3g95NLNWoXm2EiLeU9sP
s/hcLm5T3qb7a1n2tle5/chGV1HZlZlsdYytgYyzVqN3uSP5pc2oh/EtCajdr6luw+OJeARIMwcl59x1
GJMghe/ryhDA5DtsH0lbdLu80eMTJt3iTAIdyrtBWzd7XtRpaBnYLTBxCE7xnr056zfk6Fm90vs
7lWFJcsXUzJPDxbSnfx4uvncnnlVQTgrw3zswxlennDe/979qvffvNb5haqv8HO+Dtsjm/z7L2gp
H5p9QIdOXhMwQM1cTZG3UgDhSyINoP825rIVI/jIegdsTfSkFercxjXfxfxSHPRULZcNL1ebF4+Z9
DbJqJfFs4oGS7/HMZ5DRr9L0fYar78BmWzAKYD7RTbsakExKYp0JG8UNoR+EKKrdo4tCcsF5ECPV
QbdrxDBUqp3maoAq3smLaf+aEB3c64Ko2+wUSZ8HuCCi3xtjVqTFpYWqZRH9zihYrEoEpsQ1+7tr
NM259ZQns+AEmGDac+VoLLQzuGIAIyHGEPS98YCxCjyVYqQx2vFPxESwygdRLGhf8N/fM8rGJM05
LTEgapgZoch+HL47+zGHOWo/ULxLEcgGxSHEUC6KdpDU4LwPWN9Rgz1ymLirprck77XnjZNs03+o

yH0nfDi7DiOmxpaeEYd6YyA+/Md7/rPn8x++R03f8/S8vyaOez9WNGumVhwp86coqWFA5yv2WpTk
NIpqVAdmZlUURNF7SIUppYwLhhZj/PZNtunSelgJNXjOMoeJeOFzqGlbURkNw/QBLOeoyF7WaiZ0
hxRwoZiLy6zYZYp4fe1SnnO39h7QCJRCIZDATAAAG/4saCgpDdLut0jfeU4TlvDqLUE5HTXS1ONR
kyFO+nBuz5BA/b28Jg5zhNtK6X9izWq1RdkOF9U1vmXeZ9zcI4Meu1VNiSbtkCKNjGEhA65OCKZQ
qlEo1FfvKXNoe3mliqHsgUa7/Q4rG0qKUNMu05Q1BurMUf2tn6pzuFxpnbvZuabvoD8R6GTDfodz
h8chOfsSc0tst1TruNvm2Y0icddAKEzIQgv79szlqelwGQ5T1I732WImDrXJ3vdKbEDP389y7LP0
apVrZ3X/tvf+Uff+/t/55PJb3znK3/adtwemfYvYIPQhB7OPffeRjff9DVSzRTdc6EzgidjAxAwf
aJyRkcBHsxMqXtD6dGucTjq84IfsnkFoeCJ47CjWFO2z8vFO+yBMAfjAo4Hi+HcNTMDyspFqYaar
i3AfeUVDen3GloAy/R8AUXEwi0caQ4nxQmcaSoXVTDXFWA8B51T3jxQAV9dPUutrXUqCoNGRziW9
++/WmCRCHHjsEcebySo8IIRMmxv0JgzgU57nSbg7w5j0REqeKCx4Qe/vtCYo9G4L/WsMhuvbXo0g
UhWash8MQAcaMlgsCEIJxK+4/Z5ji/EMi6KAbad8TWljjuuY41EABXvufHYMO1ThVLtft6L+qadS
4NBn/NKjWxKEL5nrE9rNfizx+Cyf4yPVEu4fNZryPiSem7f/b5b4//0PV/9Hx996P6ak8v9SJPYZ
trWNp099ygdPnxMeJuUjkwqvSFDqtGZnmLJZtoyJsbmIFRlW6JdahQ4fE0KZMmstxpbS1Gk0u0A
rh3ldJ4xsaKcNkUP15TIY2EwV+Rk0nBC+0ZiSMxtYNcNSQFrdRarNOBJLIY5qwxJ2BGtJnYUDutD
WqdfZg9ZlHGyxb37SeH/z0ebFDY3SivV6VSc14QWONBX4Ac/cFEvDDCZIO9HwptyHs9NyeFI/R80
YPNFRZoPBxQsVynHuex6XBCIUjPefOAYDYG2NNI6eKAdQKbgzD88+snjmVYqZH5xZKAs9TEchoZ0
WTT8bw7eUM7xTlwEI87KQSwzFx5RhD+rF3XX54Z6xdvY7sh8T9krEQr4HU//Ltf1Pm+4z9+dDK/s
PQzvI7+YPPS5Fh5ujcuXPUGwwo0WGL4IEF25vNSNmcolixFrdTSQZzUTYU5y8Q041T16jxh63Q
naRF2genpQ9ZyFHFhu06S15jV2uYlNhGDxTKOM2kUwEyd2KNVUGIi83Zvqjmv3B1BX1qfBVR EjIU
b0drj4hQwVoL+WqdYkcoI+KcbjmvuuoPLcP2ScNgURiQ9vaOE/tlgU2Qs7DDYfPo/RqOYcUrLKLc
JfzcKCRiuJfssIGRKzQYF+ZczhPs4DJYFwoobGE7SPMGXKiMIQAw4IhYT6BRsvSZqao24nC4bjn
uW693nl+duyJN4yE/bN8TgznVKmCcIlbN4z1ueCiV/hWac4Xix45Fi3/uYPfNHne/uvfDCYm5v/w
UNHr729u3U6GfGCfeTBu2k0SYT4SeFzSWvVqGpsNpWNnF6LCESZuwLNltJwnc612mwoefae fh76M
DnN10+KISivGKvQFjBDzUBIQkyWzeps8GzTudup5k02ZYRM0E/dVVtP9OQQBr1jzjsNr0hesUo5f
gxa29Tt7ojHxchafzimdm9IFlc36NyZs3Szo4txyFuQU1N6NuyJS8U639+a6Nlg0AIt3UJtCUgyg
RGOo4TP2RbxKxgseqqZSFdkwt0cTgK+l1DyBNTNwPYAMAX4lgFMG/Y7VjyJzGp9MSiUyk9YpcUPc
+TySHX/NUGURnzj7ewKgnACQfiesT7HjNbnPMkrFNI5bUxZXTLb3z/F3+D06BTrTbeubDv2M7W+
nkKOOS8685PsMFGSr/0CoCB4IZFqNmZfBOMqRZqpom79XNBx41lwlgIKyMlJYcmBsQqxk6NG5TSAQ
dgTxTLVYLhsRsaTk3bl1RXo30in4lacT7JHB9AjBmRPG2oEkeL+tryP4HEXAQMxav6vvHCYw/CKM
Or3OE9sdXu0021Tqzdgj5sJVzEZEaFsbkPYOqtF0ycYD4SdcAJML98FqB9sba1Rt901DNFgaAKtr
SIYMVb84+eiAgwjQ67d6WxJDp3zy2yoeXh3Y9jaMPH7eqMZNfYOXHJLcx/N4uD22M534vbF9HMQh
PPm4+8Z63PFSIH99fIFnTd+FnriSzje9ksfyzKwfbZWrf9Rc24xbm+e5hzSpwc+czuNacaXGdZkJ
pUoROCaVfVepkyt5Wpoik9DK83BEwtPcCSFJSERhzq7o2Z01aSOfrJv1SxFcWMyi40dDEFxrmra
DfNC5S2bKbhh4meFEowABD0JQw1tASIYZd4E2IDrClQfeUE9VqXKACOMWNUGM7yvqPU4Lx1zK8L2
TD8AueX7ImrjTpVKzUJoXGZBQ6BPc8RoH6vPxKh6gHnrJDYxJW5/Lt8zqX5uQXK511R+XN0bXrE6
fCoAq8kRSCH+9KYW4pKtfnVQqX2Cf6gH+qPovMb5x+PFUG4S+2NU6qfLLIihlI32DPWZ7+hIjd1B
AH0uetuX/ndv/wlnbvW3M+p2XCzXJ3LsLgs15RRvFOPPkATUp4wUYIu4lmV7OOVVUmtuQqNVVRnd
RFFsUukqncLI9f0p8pYU6XjCg5zGZHb/ZMnujQu50yzma6NDMNrzqh41qGJhZ9YQPrRSKnaSftHL
WwYbSDiUgnN7b9GcMYLh17I3nKb8808G6BFBC6zF+bn6cC+FZrjXDsKe3yensz/5iyDCp5BKytLV
ODn5zjPTPhix+OAPR5U01NBWLkyaJ6IfEg+n6Pmwj5+fUKVEufuSBNE9Q4RvEvD7hZxPpoWCvmwU
Cysufnix51C/c/4nA+yoY6RDYTD7R1BOFQAQCWbactbnrE+OyzyCvjhDH8oORua4qgmYqTqc9Wf6
B9Cor6g4473/SB1uh2q1eY8jolNz4H6WUZV+UPU3Vmny+eeoDBT09zGFWHqVFTrypAVA9YZKaHhK
TWOcVoYyEPCoKdq5YKMcjR7g2JS1N6k5KxTy1DNnC+jb0q8CrhaGSBIDfmKRyY9Vw6jOVSFrg6p4
H2mGxsGA/ZoA1o+8VLKlascjpucP+dp/5EbqFywqVLik5sEZCdD8j00ldmw6rUV8dz97VMCOVxZX
qGy77DHLXEYHAI MED15JNqskeTjwDgj7B1wnowUIWdhE8hTpcI5s8v/Zm/tuJ6Q1zm2m/E1BLlc7
oKbK/ydnSu+f9jeuvfcQ5/qFQqVZGm5nkGqM0Lo/SSC8GdGd/V53bp5zf/9n5/58rI+Tj/2qGlb1
tIkToqVQrHTCoa nJBeh7c6rIhZvDm5tiaLuDi/cWnokGJD1BQNiVxMkZTHbJgiXcyLGARfBMJuD
kGtxJpJcohtC6TR01QvjrSHppqrQD4Judi08izeNhUDn01OCjdaIgz+hozDGRIBWHBj6b2qIfaYF
3nz+MvIbR6kkMNXG4DBUpUyTiUmwy4NO5dkHE4KUJzHjgdGuycYsPNU6M8T8VKTYTabHAIZxw5j
AJpDnnngOmYPHvidQJQBv+OX5mXgXTPL1B14YAQJPTU5x071h8I6mUZBWihWB36hyPlG91HDZ8Nh
94PbF1+olMsN5NGfpytb3TIYcP8un/7vufUurZp7/gnHb/67TcDuTT7/vd+4HXC1geeJtMwXMD2X
sTmcFM8GW8U8qVxqdQUWYiw35JV6JUqlGcvmzQPUEfxT5IHPhnQugjUz1KjzbJrqhgvUEBStElq8
FzC4ikAHx4QEz78WoR4U6Zk8aCkhmUAvpBRN4TKgu3V8heAQKLgA+EpeBcIbkFxINHeV0SUVThuc
r6XssGgGJRbPET5xRM0AfqIc8+QnwNCN1SEcyUF9Dx76+LF4nEoBTP0M/N8fcVSkT8ne12nzF5tI
tuMi49qpZLzom6QggEjA/dUQoA+5K2mTNqMh33y+fe50hyZbiDAjmG31bj5cpfv/X382o+k5H6st
Xb+DH++XjzYiXP5Uma6E7JjVU1/rh17xvolHr/2Ha9gW7IyP5czfvf7XpOx9wTQ3/Rz3iHTtBqma
XyGw80JG9Y856n7eHEUHDmcoZI6JkXwhcGAzto5MtwCtZ64lZqtTfZCFcHOSidfE5dhgu9rXTL5I

2LHqYymiYLP1A0RhGaZgi9mGMBGg1JXdxV88Qr2c8xuo4iGdqtpCpZXhuARhgIkEGseJ3NK8h0rX
WjN0igCxnYdxYM3iBqciRE9pBKZTxMTraJmzpsaiuwTV1ngc3WE1T+fQ1W3Qjbnno6pQB82rh19X
/bIkr7ztWBGmLc7QSGV/IKIhI1HhAUxxDlmDhsTwuRSglw/SdMwaNmefzfvcH8Txsmdw53Vi5zW9
NPJKC6Wylk/7AsBW9DnNKR5YM9Yny9HoVAAdaV121aeDTWxxZtaZc6nvp5z4DGvUfYoga84E+zN
3o0i603OAXfLTgOoclByUhyMJBvb557hNZP3S1h4Lmz91KZw8irT1xLOUyTYIibQ01pp6SZ8qam4
k8SJXZACA3lQ4U0LAP+OFMF4EQhoPB76NsolJPaKIB8gjFJlpuqf8rvwfmeOdGM9LoKreQcDcmHx
XmDWSIcCbFZ5dirycT4IDnkRCOZqQVlSzroCP7Cinjt8DP+bVURo3mgYalJaJ2ELcEde+WmfKaYo
wJo8YAWxiqUTLCXIVM3fs6V7/P82ow/HHrNjtDYOJlFqhu8GWS9nbWe7Tr3WYbxV3ye06PJeJ13r
GE47IVRMEh2Wp0MDIY2581uYtCL3/buPWN9vhzv+OW/F4KWD/z4WwccmqamojHNceibY4+6wY85/
v2N/HiQF00ly9Kh49qWUa5SORyLUPC+F76Kzt/9Edo69WkxstbWKgVsnI/ffSeHkhYdv+ZaNYEzq
bHEiv1+qjmjCc4weSIUMJYKf1GwgaSEcCvp50oLB7qswmOcKWEooIfQ1oljIRizwkAxTzjWfQrvu
qBkuZLLilGue90Je9XyNV9NdrkuBRoT0y+OK5Q0JlupxZGCC2ecIBw3ZWA85fxz1BpQa2eN6nNNz
j2XObJQGwJ0ZgxMFMGLs8fdWb+Aa0rSycTw2IBz1eWk2Fgiyy9FKX/IeNxLeUOMOVqJQW8a9trps
LV5ju/P33nF/F38nA3+XX8y6Af8mZJyuZoZk4wmHIp9WVyzA5ueb2jPWfeLzpx/97+qGfe8cU/
VPkBWYCunJvDeMhXvZtNpj95bL3rsEw+TM373+f43ZUHgkdUg7N4JEGJwmxmPGYw0uvSftbW3Rgk
b1Pqc4hXpm90UiG01MJSUkXmLxFPijUocJbqNsxnNumWoEc4HRSbY00VbKNMusjPVoOdcEomChhr
SzdpCAR52pau7U1/t2YFztY8es3voncckOTi+feQzuJjBeItGPmJbIB2K4rda6EvWDOc8grDmjIn
rLV7cgmsW/ffo4c0EcZ06i3TZWlw+TGUVyozMW+4fQ51045hWKUKzUgMz3meGKITy/fdjppbo8DM
Hm7Lli9+TTRJS+Xvy1Nk9UsMwaAO1kcG2fJJAuNIqcoE4rRH+ZQvXrwuj1jfb4epqWkINkkriGRj
8QwqnHKEHp8qvIi+s5JYt7GEZwD6OBo2KJSZR+1L5/mRT1SzIC8uGEoYRgJeL3LoSSQSRF/paRIN
udshkyaBYrU2xSiFyXpmIDhXrViDC0tiT4pwmEC0J0UUAL5HSq9ZqYEnxQyihTbfqpy4Uy/XgEqr
FnbCAWx7s4Zarz8W8kuNhTFi63CU7wPvPAUBmk5HkfnnlCPptCGJYTsPhXBVoh8mD31uVofofEk1
umiY1dfTfninHjiUn0+iOP4Qntr+x7TdB5PI2ene/GhESpDnBtHSTwJMa/ICcgkisYTKtYndmYkr
pvrQ6IzHLAHHBdHXR6aGOKktf2FFCircQWudCtU5+bzHX/eOPWN9vh23/Op30Ve+6zcVpWiagg9kk
Rfsg7xq7zGUqvoil5R3JUkUh4PRb2dp+uo856mgOj1w8tW09sS9wiMk3MUuqEMHnPO1JBxEUUXCT
zY2UZbDYDUbg5Mrc+6nRKSibWBGUSnSBOAa2zt1JoQRgWM3m6hZUglzpbcb6r6bWm8qVhBF+anq9
aICHUtlmDeP9jZ1t05T8+VvIa86TwaubQ1JFLxL5muqdgSgU22YvG4qCinXkHENVEiy0UNqvP77
Fx4nPzf9WK6/daPUXNhhYzL25SNO3TVVSfi0vyBbm3u4K1G5rw/TuPHJ6NO6Nq2kbB9Go5t8fWY0
XhgXNhPLJsJ4p3E8csxb0QBb2RxlDhy04TM1cU3LYMEbMBm9xeVzWC5zC0YM9Y/wmHGCpKMHEI0
rWEd/w1NoG2oeBDL+f19PpoEtzJK+RTbBiv8fKlw/3tS4Jz9SocRrKXqTSXlbGmHEpCSNnwBX20s
7PDax3D2HLS2rBshAiZiW4xXZu9V04EkoU61DRV+8ZQquGmxhaL8U5bPYziUzLT3SEFGckTfO9Uh
HgirVecUC3/Bh12wLJm3/5vyB/8ShZecVYL+fLbGVWBZPLYPFAf9GFRqap6DXlmg1o3q2js8hN
29AcXNA5y6e47w5T/1BSMXAoE/f+RBdbEXx15eLW/sOXX9Xrdj4zKS31RuvBWYoscmhHThsdy0zX
zKTZGIYpmewgabxZJjG8U7iF6tpqbmUublcrw9LMSOcqy1yDm9S69LjEuYDPQYe4z1jfr4fnOvx/
xMrTZIz4Bhkh30tL+VjbAnv40UN+bYf/tnBvO/FOB70ZBh8k70q2h/VwlHe8XNksetx2hzy2gp9B
CB8v9eiufllDn0jJVimCigMMgIZ3EAXS6A9AughNHCsnPQoZZYOXR+HNMrJEMHnDNIg8KSgMENVG
F6WttfckaZKJSLaJWHI788edfMCzd3wtVQ8cC050PMxNdk4zmtobDEpqtUpZHKqLGdAaZ2fn6RsK
MGYco4r86jgO+52+2LQnKZTHtIlqUUPPnCeVhaao6X141tueXHUefxTiVeoYXgdu5BRKFwi7fULq
IpR008Y2GiK84eyUn0hW7j6pZxNWFky6mZoNyGHH7cvci7co/bqOel709Um/+zynrE+n4/XvueP6
a9//J81aRq32WBzvP4e4B//Ha/sBi/cBTbaY4Vy/WXxJKhE4ZC8POduoza5nk++75NTniOHF1eh2
yXAEaNJSSyZw2JNQC36/RjPQbhhras+JdoqMzk2ndFSLJpO5VYAbLCXbiNQWgHet/mbFqoWjRKgiA
dyrTSeS2VDIUYC0bNjZpMrxm6h8/EzyuUlkKgiqDvqHFi8tJaCNKW7pKCICiUNSwEPTCoKyD4Hr
8znh7LcVmdEo9FIKEcXV5Zp36Gj106NacjBRKu15jUbB80APSUn7rbWXX5CHhlqVCZz1L2qgUOv
V0ASmpL4FXKEn4ejJ/DYhq31ljobstsLCrXUEBpL5QsBiKGPWP9Mj1WP/TrhoSHpk9LX/fOzws7Z
GNBoArGLrARC0d9Rkadl/Y+ly98Je/5L+Ycs4TQUiQdeHE1lw7LLKjNnoaCkTailCpbZK6tiiFib
CzkXE6mUrKpUpuaqMk0telUWEoKQvC0GBQwzNlsqtKj0bKMgt1jT4fvoXeTaIEqkfFIBDAfYT503
KMxbxSlw9dT84avIp8XOhgKDeEBTmgGk8Z7pkpXlrSXFwk10Jec2ZI+sCV9XSdXpMm4z7m5I2p0j
546S9f8BIKLKJtWbE6wFx9Qr9W2W61C/fy5i8cvXN6u3fCG7w70/v2vpSg+cawrfVLk562trYxDX
fa4Y9p89A4JcZEvWxLquvy5MAA/5Mi5xAZbpXx+IrjjoL9JX/Gu/7RnrF9ux8aH/z9zMm6Zhf3XZ
bml67MZhcPnOV7/Yx+Q33/gPa9RbIhpluR/XJ0rVb7B93IvTJK00u/36zCqUuWFXI60K80VDi+vp
ygIBa0D9v5CoSDMCqDyNMTLjYSD19AiyKJKL19V3xOsE45AEWOpDqdRptTqALQ3FAIKHMDyb4AhE
pWzQZ0DbPUYgheBKzY6LHBQeI45BC/ML1HzRa8lrzrHUb3q3dDioFCpOtI0cqkHOaCoCabylySU
nSToDox1Lgh4H2mmkvd6bQ4YligztYWDQcBtfnhr94FJP92Ln1+srC3M2FSuW+n3/393zsR3/t9
wfVan1275eez12HPbN88tG556+MzVt+z41GnWJx3/VO9cTXkVdbzvJzB/9RMP8Hf+Ytlu24gCFav
KBfzGHf1+a83JHMcoN269KyosI0ObemafneJvQ5/EIZ6XJyOfKg/cJhHUBOLm+CZV8ZJ8JeFJowW
WNopTYBLYBBwnSF7HvaF82m2jSpogeVohKYKOKAPf1ESM9BIYOcVBgVwKkbBILDDQZ9moz4Kz+cY
oWaN75R6E5RyEK/1TSsKyhvMlWFBveRJoarA5syvRqKExk9XnAbA05oOnmZbrHcPD18/32ixN5oV

mlhgb11EBNvZOWF89Tr9YoPn738wq12701uoXLy1977ntzeqtzzrJ/zSAdy87X5uLeZi3q7mx2H
7u9axRKEY1Dq1731TODPffr3zNouG0EiZfHkgR9SNfLgyZwHzuca8vF6lvZWo511jfeWT3zCohFA
erWKC/QsZe/kexiTYHuEwVe8HyMzuXpoUceY48HSUJHIHUp2PLZC6c2DEDTlwAza6sWS0bagAXDq
9TgIg5J4U219ymyHyga4bmReFQwKoAkGxXamHnlgDJQJUXPF9XexZvfRIWF/WRxTi158Yw3LrvCY
PXXRCmgpzE2gJHksdDugaGi56qEhQyFvuINZnV1nWqNOoWpSQ/ddy8dvvoE+dW6fM6V/fto4zLZ6
+vrjVK1/Kq11a32cHxx9NPv+ZFH/t3P/Ptwz1j3jicdiZnaWX/zSH7pxMnx6v2XHC9/2rTs11NdH
m3f+T+SzM3x2owtGq77RnE5c5KYHdbET6OwZHvewXIS3hhPxq+IErORpYa5sXrqFaAoWd5/FTWWD
tHSiZeTC35bNkQYXTyBuvhYPJhlj+ns2XNUsByqVEpKc2am5apY723LevIsrqXlBQyF90WRyBQK0
UjA+1mi9FYT0V01BHSfSplJvTfU3WCsMRurxbnz0ivfTIXlw8pQVSVJGyYKSaaIP8WpWguDNzycD
quDdynKdFEJm4dq10ixC31fCMzwdzxqVvp/gfOUr6coxfddDOH+QH12y01BXv+AtVqNb4uj2+yf
SBOjK/v9frx3bff+sfvevv/9uj3/psfCK5+wcuyPWPd09QNqSxbw3N3lw3Le2nm1b56tPHIRWPr9
GBmmZ1kPB7Z9f3JaOOb1dXCkY28bPJUGnG4VE7ncyn8bjqeIUR+5VKMhoeGAWHHbj8qFr6cC1L
6PqynFyK/OC6snCUKhS5D0BfnBNXvBbVPBdJZFR8MXRztmuELOpVegsQ8+3ZpqqVED21libezqSZC
14iwbNJeJOM+WoiI3DITxNQusR8DZESc0Z0of/S11C+uV8KQ8bMUBXCSbSpDKViB5JUKJulwpA4H
atTo3rwoMBJC80MNg2JwxX16c52i+6570GCRm+5i5doa32LvuWt30gPP36Gw+AJ9dodSngDQT90M
Apyw2H/0Hg8/udJZnj2TN/8ee//9v3ffVr1/ov+7o3p3vGuneAlc/KH3xJPDz3aT8Lei/OLoumJ
B4G6bgPGNwk3j4H+J7L7sAJglEuisKCW2i4WZA6nH9ytupm496ON+KwGIwLh1/wNbRw9CQVa3PkN
fdJrpYihMW85hScICrme9FirRR9qjbmKRx22DjnFNUK/94SIWXQmijVNLOGyxULBAwkIcV6aAOYk
MYzMS41/2FQBGhphMbnEkE+ZZILq3NkVDv5SiosHiQ757NX9ZRXh/xkpuZe1Z6h2kMzoS8YPopUu
h0iU7G2GgYQwSqwVGhqTyizf/wjf0OD0OQcOaDa/DI9+ugj9M1P30+Hjx4S/uBL5y9JfxS5dMgWz
e9TNCzrmMcJPHvsxuOnzv1lp/V7n7zr9ls2v+fHfiHaM9bn+THkhTy+/HjVzCW0SiY5Dh3RoLNS0
0kyEDO4eQ4PcxQGQ0tQPuwlJ9EFyy8to98nWqRWeZhm999A+cYh8gsosuTjr8wp7C5oPiNDqDclT
BQCQRXu4E1L1TL1ti9RYY4Nlg07r4nMhGM4VWLFouc9CyZhihXeWwmjPWroJON412+X9G5UBQvC
YwUrwPnEED9/PzqsRdR7dhJcgtVxZYoo3QWv1wZp/RkMz3ELiF5osNhZQ1lqCKTDMnbirFCzoGCm
u3I7zGocNutt9L6xjY/RSgnmgvztLq+IwiuCacKCyv75T7sbK1LAatSq/IGZuQm4+EBzoULk3xuc
RyZh9fWtz/8Q9/+zU/kipXBT/3K+9I9Y32eHvuuf2Xv4UuPrmRucb9tgDLitCb9TYsc3zITXyHAY
3soizpXnuevKreEYpNBeahTnKdcfUVYC410TA7nfoXGPlm4MMY4UYUWtFmMSAs8papIBJTRzuZFK
uVrYpSxNmYY1JXYMuKWCjOhwuEq1TclqC2OT7opKAgFangdfVSYdaLaKKIS16oh9NhUPN/FA8epe
v3L+LM0RL9VEEemYnxBzjs12ClvVapJl0BLNrR6HNxuqnvAln6eMnglwgitQsP30/3PXS06nOLg
o6aBCEVSxVau3yBDh65mnrDCY3PrYkOK/rKIEZLgVoVbBB85MvlhVKpVBgG4cpoFB4PJpMPuv3g7
v/9m9681mw2g//3N3873TPW5/Bxx60fdzfW125u72xcvbF6+aQRDQ+P+v3F1YVmvUCjw4eufQE9e
M+dyaC7nRw9fGRSn9vX2964NMyGm3T1C15SdvxyEXxDaflkskcFltepz0mPEbmbJ5bEz1Fucp5V1
H6kV11Qbx5PZNEV68tiqMgJoduaomgreXlBhSkTNJnqh4o6t7D0J5KLokIr8ArO+WAU4FnK7Ex6q
jA0y7Z2q9jaQITtUAwnU8Yk54u0EtzUSysqF68+L6CHPH+1XV8po2v2fkl91ZmVWruM5ilwRZwoD
4+815Br1YPxhlJMn07tpBEkLnwZ57uVvSpy1f6FVWRWSlSvlWXYAKgmTBqBra5s+OgxT/h6JzE41
SbUmKvT6uXLlHRTe2trCzCqXLVarXW73aOmad5p2/Yto9Howf/zW79xc/+hA+G7f/aX0j1jfy4cP
/+TP/qyc2dOv5Hzylfbjv1K27CtixcvCiLnqmOHpDKZdx2oHdGZRx+mhYUla/nAUVBzcc5p+r6XS
5aP3sxOwaSd7TVr7fJ5Cx509eIpgfJhJrJcrVAQxXThiYfpDd/yXVQuL6nBb7+kmRhiUR/32XDFO
wlWN9H5YboLrM/UgPdU5gKyEKNBn/x8VRFsJwpVBP1WdShvKcrp4GdCzmSPgMM095WqsHi/KeWog
hKaiWLTv2AFzm3zBWq+7HvUaixK5Zc0Hako2SA8BldTpnq2FmmZDan6KsJvcZJ4Up7UkAeETGAX
M3UvV9RqOPvgXu+4+O3y3VN+N7s9EfUZ8951aFl8ngT7LS3pNUEZNYkjGvJCr2EBoMt/r1Lhw4fo
8sXL/D3AahwcqNxuMB5b4X/REcz79ha2vjrtFocJvJ2I98/796y9bivv0w2mTPWJ+Fx1/+Z+6d
9728bc//vAj38/h1bWupzRb2u0dyvOO3Wu3qFAqU8Qh3aA7oMHwIudIQ15oAdWKPl3LRLwu+OSsD
q2Vg4esOOzTE4+do94oZW8woXjzHHV2dtgDhHTk+DX08JnzwhD4lre/m5ycCmFdv6ZIrzMVj1YX9
onor3KyoRiZkrpSHk5EpSSn82SWVUSLZWTNUSLFULCKZy0bGCdkHgxe1KaIHqs5UozMZqThhPJ6N
VurPLly2PDSCTYM5LxgkGDjn7/xtVRZPCB4WhGGylR7JgXNWaYGyaQCTdNqtNKSyaTyG2H6TBGra
aGrXUNFJTMV3BYQRQCJN1cv0r1LlZUnsSH3MeGwfjgcky+V75IM3U+iRISiAw6RW/0BdVo94anK0
nNCxoa5WFKht83nKRocX/Pr6vycq3hzeGm73b7n4OEjtd6yurj7wg9/+tvUXvOAFwTt+4IezPWN91
hw/95PvPXnnJ2/9b3nLOQ1dGd5LyAbW667LIp+fx+TwdKqYFMg4iWr+8SqNgSI1GQ42EsWf2dmka
ukgHTnxYor4tR/86G3UGiF0dGjYb1G1nKeVg0fEYg6/5e/o5le9jr76j9SGPGB8wUeVzwqVAmv8
mJzUQwCI2QIIo2pWjiHxjBoVHVNW4HvQQkq+V2c6fz00zOpigkURgevozynMiBD690IO3+WaQyRw
kQYur1jikSjucv6AAM2VWFr7oZXU+3wCc63vRnIQT1PTdKocNpQfE+oYKXK8JNYDZ1LnlolcxM1A
FsNE2hRarSSUpoga5fP+rEPf4S6bJgWPDWxChyCGzYsCYEtDu0dP0+5oiM913K1Rpvb2xyx10We9
4c9qmRlQXTlfEedhz96GE1s23SKfiGf4889b3vuifPnz78iHI8/4/n+radPPfqXpKag9oz1mT5+9
r0/+v23fPhDP8uL3h+nA9q3/yBdPH9JaCrnlpY5VA1ojY2zVm9Sayeg9dU1KYQsrxygkMPZSrFIr
7rpBB3at0zNxSxaXFunj992G40Sn7Z3tqm1vk5Hrz5GhXKDLp89DbJO+o7vftC1ltgjjwWhy+VkiC
8wtFhFaNMLnqxkF0bIQhUZTsShgmBsJJ0JlTM8AfA4j1uXgGauo8s+GhMEq5LRL14coYW6KHY7W+K

jYEkbNACm1kM6A/WjowmJimUpI21U/eTM1jJ2V8DcacCkdxJsLF0ywVea6iGJ8aqqgZlgAg7S1V/V
TR5dNUY92A6iWPKMK/iKsZQAT/z4x+/RRGzaT1XiSrsWGREWlubvNnZ4mEX1+Zo8fhxevThz9DRw
wdoY3uHOOQVqhgoxUMpIQgmF9HItCMiSVsrpz/2kaaFNvbLfBfzfN9Os5/i1dsbm6+/id/8J232
5b1B+/5hV8P9ozlGTp+9Rd//kc+ffttP9vv9ahYKgmEbjAeUBAOZMokDHPShMHIALMePctaLditw
Q+0PF+iV7zwBF178npZZH/7oY/QPfc+RKXqEnVbpyUH/YqvfZ30P7dWz9D+g0fp9W/+JmGc99BLF
GW3XeNBI98rlgTwQIknIXBKcnyQKak4NTEDjyve0ZFQmrReDM5TnVum5eX9wMequdWMZ1KQqMoqI
7WVQUyBv+ZUbiOe6eXatxoQS7ZMQSiJE+b3rVzzUmocfyGlbKjTPcGYKstpXRsrWE4TTSyuK8Cpl
pZACKzH9Wbq8ShEGUoIy9Bka6IGH0/PX3e2NuJUsd0iK+iB4ev0+bnDTnQCS8XraBgEvHnHLCXN
8lrrnshbaxeomuPH6Vz5y/yz4cUYridFOoLITOY/VHAq9frsvGlouKe2Lxx2Pmc1+Rdpen5pWptX
v/1URD+H//6n3/9add3f+h97/+LtT1jfrqPP/r9/3LD7R/76E/1Wm0xRoHKcXh3gXPJer0qi3pjY
4sKnAvhuHzpHP+8yYukRBEbE2opLzh+kE6+6AYhJvurv/0onblwWZTYet0WHTx6XCqxn/nkLTQ/1
6TXfv0b6YUvfQVNgq6wxMP92Y4vEy8wONsxJaeiNBCZhhwVlcloVqXFws00bYrriNQqUsjhaOQDYJ
1PUj0nHr7207rn949rHKdCDGAsowqBxI4ZkkUW7UhsyHqfeTCnLqcqVeHMPZHGfWb/25VReOSqhr
2rF6F7PTKDK1EoAqFjFipxNvHMq9xg6qpTtjsGRrvyKI4WlBtzlfl3pRnUZx6dudvZ29ImZuObuOe
aOx9PCgpXRpZfgcI4N8Qef5b1DijTc+fZqo5QLle21aXmzS/pUFmeXF/Gs44QB7okLx3vYWDbsdd
S+zWEJph99rMjFsRDHbly9XecMoLi7tW+CN/CVW7L7xbW96/WX2yt/LYfInfv2//GGyZ6z/i4+Pf
ehDP8a7sBXGCKwQc2iGBQMP2u8Pqc75qON0xRByhZzSNjUED0TVnElveM3N9KIXv4QefPQx+oM/+
mPyiw06fOQ450pFETt+5IF75byvfN1L6U3f/DbOkwoyqmZYvhiqWvCWVElR7VRGN1SKZn5JFpbi3
CXpO6ZT1kA9ZWYZecWvIgs/msEK4R2LpZp4VAAfLMvYlXjEaBty3MwRozGMRKrAwgSBB3JGzJDqi
m9iJpIX+odPumHfNeTXaqLnKgUh8AojnzXMGUY/1ZIYsr1IYpmUcNUSEkwxXGmP9+uWLPicjIE+
yu6qZY968t+6o7bZegd90Z6xxNJCiRUR4huszeGbi02BIvnb6vXp+1WQo0KP2p5snNVvj8T4VFyX
VeOTYtFFTFkOozHjVUqCHyuIKSddg91KBV9ZJm9zfkvNpjhsF/h8LjSt/t/yV+73/LG15zhc367n
ys88Vt/+P50z1ifckP9oDvo9l6fce4SS/6kjMAU3iCvE61fusz8Zi0BDrbm3Tt1VfTmHdlzxjTN
77ha+nYNVfTn/zJn9Dtn7qfcsUS1ZtN+eOfOn2W+p2uLLx3fOd30Ute+nJenKEUVWADDha3FmSyL
BWigljbSC2l5cqeHDhf0pVUpet6hecDgslQRBFGLhK+BYDzN7aFC6kYNjhRwNLikadV4lGcVhp5
xUToUAPFYBfxH+lnKMYlRa3Mvn1pcUjVLv6Rgp51UacnxfMprpPILAVq/BaCyidontWCTJ9VTWXS
mo2V7CFphp2hxfVvE9T8AM4mxCaW6JIZ+kKtko/7r/7PsUNxZsaJ5SSe0aRQjslhv6bhbH0Ygcc9
cQapWWkXRoHaa1ttIQOxrANebZyeaPM5TxBSmXwzBEGoib8XoGEyaat8mbZjCTPTqnb6Uu4nHD+K
xQ5iZnn98jz+lkKx6M7B4NB71++/mvuYsP9IX6c+50/+UC2Z6xPwXFx9eKBIaj8ELOYNFYpshNY
pDMJawzJbza2lwXycEjx45RIe9Sndf6a171lbSwvEQf+Nsp0aWtMR04fh0VyhU6d/YsRv6+KAZ7Z
KVB7/qBn6R9R46qciwbnyCBrtCQy7RGapoEkre6BZ9cNnpUWg14p0gpu6XZVLfTt2qUCfBA04iai
ml/PBoQn4BG/R2prvpYkK5WOEcRB5y8/z97XwJt2VmVuc9w77tvqCGpSqUc0IIPdKQoAtFUEREY
6OgtLpaBAUVFIwCYUEnC8WBOLVTu7BFRWjtdliCtiyW7QCiIizkHitzyFippMZXB7rTmfr/vr33f
86t0L1W9zL0SoUKtap479W9557z7/n7vh2urU/kk1GQCS908bOUEX5+xymydM5F4c8zSa+D4RUrK
7Kxthpq9vvklHmukIZAipzRCCMYzwOryveg3EiXS+ncSVk0nQAF8ab9b9nv8VcMeyUJZP+Hx7b
rouGGSms9tKx1VoCiX1METajBjlylqTeF+ZY4MBglmkFEAFaMDh4EJ10/k00JahQRyFpYteeY8dY
kWCPxB/242B6FmgCchnl3CfsxRpASz1lvDCW7KsPj2cp1cF4z8SUuXPhkzga8Fwn/r9P/vL5ivG+
v/46/D+AyD6BNV1hTqHRFewrllTs03YpFlbWw7p1BbpN5vkG155qZx0yk75m7/9tNz32H7C2zaGp
ew/eAdX3W9dWpRvenlL5Ed+9HLO+nTDqeJ2HRmko5mU1DBucQsHBYAk1HxknjRjPfwmtbYPLXJN
emDXq9MK6KQVMOopqES9xtebzoakxg+HyIJrKCMmrUupNKUtFRZFGtWURw4PMG54HC2XfwK2XTK2
WpIKRUXwz0ZMbrh2tdDhvFkOKgnnXYOpU98ZWT0QYnORuvE9uHUpYmfpZzrlpa90PkwW7AIhiabt
NrD3LKe99lAuv2220JNantisUoSguLBqYDXC3AEYIEtK0I2HQog9+RaSstoz9wj4rRM21jNE/t01
bEkTLWpMUVkVTWTbcBXDkfrIVvp6cy6zFgm4GGryuD/rkq3hDR/SzEpFzDPi+8ZpuPVN3zHt94Ts
qV3zs3NP/iRP/94+RVj/b8x1kPLKwUecKOrCXFY/CGLpXPTSa2SnOHQDMcjed6Z28kZIS3+zKf/S
T5344OyHOqiKRslKVOo551+slx++Y/LhRddom2d4NWxQ7WsFIIdL5XrUkfd4mQLaxUYjTD1BHKdif
qo7ZcQWHIOWRtX7hEybdLBIORYcaAiIQb2B2UHjzaaxjNAwAepHdDRT2zLhHqJmQNmUedNTnaPw9
AbbTiEHtRdqZWgw4X0B21NsLgx7TWe64b2P7HtMVg49Kwdf9LWq6wSGD1DMkHhJmjhrdVoeZWLq
126bGkvFW6AG8ZyZ46KdGcr692QXWDEgp+951+ukbn5BSnXR0p2D5kBI5w1rdiMorNVQ61N/X+OZ
Acto7001ZNI7yEzXqTVm0s5aFhd5wPnatlAmmTU7sp6bF+ILeX+GpND3RubIAVZr6yXrjOLEH9t
oR/f3r487o02xh+72WvXg5//52QHfz3kM2tfeiP/rj5irH+H34dObR/pZmWJERcKwgd7rJkrYL6b
jIeyuKmBQllintjibzwwDPk61/+1WR83HH/XgajjWDAC/OL3D7+zS/7Knnne95rC3vVi40UjddPO
rUnoiijnhgDTFzr7gxRLH/VTMIpyvE6qWINuM5g1KaRul3iwVW0QpWLBOWtDGLtI76DQgOZJo2il4
fKyHN2/19QKJY50OMKpWUmyIQULCaT/g5D+n/L1r5deuHbuoQ11eRVSv3RF1YGF64ZsTDDy8WiNM
MHJ+oo8cOu/yrkXf40MFjbr2dVJu8DZTIBZxf92CXiTrkMHhS5NPfOo+XVc68En94WS5QDpdhrlC
n5GUPi4vqMGBKJkDUvwh9EFOT/u5a18jQE2tLmktT9+Hq/H+hZoggu2JAJE/N9w9SW4wkYZVNKBS
uuUtbAnUFqpUtUqdt7rJ0yp3VFSd7mut2RnsmVaTXeFUuQ3Q2b3s8MQBd70um+/MWRh73/eeS948
H0/98FnPOo+6zSYhmvr72STj1IhmvauryhWTYub9SFUIGdLSGuX5IstC/LSF58njz26X6677UE5u

j6WtdWjcurOk+X0k5bkZ/7jT8p7r7pKBgPdPUo9o6o2RftUSdrlWBE6ENVGPbm4SRa3nihLJ24Px
rpElXc0e9AY6gXjBB+ULBymi6rakA/UmJA+Y70j0vjJxhqVBE1Vw3rDEA0f3nNDiCRzgmVMvVz3r
3K0mal0KAES7HLqDtVt174q1Mub6BxwENeWn5JssCB7770xvP6qPHrnNfHQw6nhd5YvyGS4IQ/fd
r0Mg+HWNqo3tXaZReLBt+GO/p1RNY0RDh221OCNXht6YwnO89abbg6JRszRS2nC2t6UWgjOaJPdE
5WhU0MFBwFu0NMU2NWGRloYiyH1RW2gBALFD8Ngq6R93cacK94PURXosliOiDpIX0yN51w0Emtao
N1A3YMjwPUXALNULcySXyvQHCT606rePhmNTx+Px69fFWV2+7u49tz125U+8/fq7br01+4qx2q/3X
f6OU4vx5J2V3TzL1oSgICzx7c2RegUvDF7pmbtCLbftZLn1rkfk4Kf1Cmdv3rwxX3PJOXL1r/yqv
Owbv9mUFGpiWxsbs6TW8SWyCF1ZpIw9lQCFN0dqJ7mOL8hnDVEcurTo0iKS9RcWKb3Zn1tkTQsGT
hb+XiY9Rn4cFERpHCaN3KWsLx8kXrkPAe6ksfpQ2H3VFE0PJbEhNzhp0sOsM2XT6ucodZd04ie7mh
J2ncUnzt1P04riHRhY0OupdXE8/n2dNu/euW2T18H41RBbfeujTug2pnnKm1vQBtDIx2CDVJHRbs
+oFI1VLdKZ6W6hXAU/0g1+A6I7UHfU/HE9ay8LcQE7cslkWF+YYUXsc5ft0/S31RroOOVn1/ujcG
s0wNBjxJxxsXUVVWZ2ujKrV6H2VSbUZcdQEeeI3gmIpClTZ8Kx+wycr9bVT6NdNUnBMH+3wQFOAE
43U4TNUGQ0nuv7u3feSP/zdD3/2njvvSL5irOHX2urqz42mk3mvMSrz2LjxAIUDEA61hclESdIX7
94te/bsYbo72hjKQp7KG77nu+Rt118pm7ZsZYeVq70j+qMKSSiJz02tqAhGvXnLZrkPKwbo1Wij
7gqIvw8IiciKldGAN8aULJQ3TI0goAkynocUYyDQU83lrXezXvsYroiA4z38KNfZLcXXdI5pK7WY
CHg14yHs9BaeaPbdr80pGzKloGjwmwzD/UyJHbTtjPD30P0MmUKjfQKV+yHOnNurhcygiWexv0P7
pEjTz1G44hjoMZmqK1BCjONpExDo9xLHXs+NIJKU9y6l1LE7c4779Qo5wguOLBMRje16hWztx2+v
ik4tx0nniAnbt7EiKoGl+r75ZkR3xv+07xeYfhk/MmZcaZNrjgicyfT9BTt1bR1rGZMZZshiGZpm
pFZhE6s4VaJtRlFtZfpuGz3rcFBcT34PQ73+cDhA1/3Jx/76FXPeWP9we/+zouC0b2ZnrEyLmXqN
UfwtCFSCGFvOAJ79+4NqW105MveQ4dlLaTJFzLvLPmpD/yUvPrbX6c1jBYpOjPkCkXlfdINLqzFj
1TPQRDBa069e4986INXscva2KLiYTTUgkhn2eltDK0iR6oFmy3QBB6vHqaheiqY2FJjLKPCe06Lk
Sw/+UUAUy8YXb+nIH9Q69JGjbG2xg8+/9zmbTJ30k4pRhu66mKq2OKFE0/ma0NmtAoRH8aTWzoNU
AL+3qNsywIJDxkdSV/2P/KALO9/ohVhS7XzzGhVaR2axE5wK9YmndTXIXa0pY7s3y9PPPEkI2R1h
xkGg5QSjZ2UTeyEdaPXpWgkDYLzWsA29DyPo6A4IrL3QqSeTetvBHHDtPUPDGnkBndT8sFFKPsx
FeNRds0dp85amqsJxHufWXD76jgmJTkHszJn918VOq1xFm0y+LQaCfT3r59e9/9m7/wwe3P6QbTx
urGL0GPL40KayMgdY1/hgeFcQ27nOEBLoQIVYeUanjoGhzTt7xGXnHZ65ge4/A5oF4XIGM4Hx58S
IGSVGsidF3TxcRJ/J4OHh//zeflk/99V+ze7zt139Tfux97wOcgSk1ZU8SleTEvlHUt5jPT1ZXZ
DRaYWSYW9jMKOtpL8YbfnjhoK/ufYz16GJ/SSVPjPkCBQhIrwD9B8I44kfTyziKmAvRHPTbWnDFB
k9FXC5TNjgCbjjPmC2kaRFq4UWpsorXWXG+qdh2pKv7HriLc8ztp57dRhw07hoQARTs33TqWVeNQ
DOp5hY7XQZ83IPcdP11wRazRqMcXW4bfbEZRS6usyEaPfj+frqownGOqfZAUkaltR8Duhyg6UjN
prJ2zDK12U/GaCTJre5rTo4NiBtURfhm6nOrguSBFKKBtRJQXqivldtvdDWWdRJ3ZFhFWnVMzL+G
22Yhc9YFNsfuOfetwIA95w01n9/2be8fDwavsbn0qZMpU5s9QM9rCiSCWnSFOnuYE6RQuWqvOvt7
5Dnv/jlCgDALnCrNcHjhLekTEpFyL2mdEj1bJTwwAMPyKf/8XPy2X/8vAzH6ChqB/K/fvx/yPYd2
+R73vIj7BLjNRpGma1BJ1h7sbain5j6uX3+HDmpUwVycFUJtWGEFYgQ9c5BDkh0hgiPD0cCA4B+A
pJqaeBNDBYYIExyg5wPuE+ldBOSmGYZhyo6xpbB51dQUJiPhicBfSaMh2doBYLYe6Je++SYbjuU
8+9UDOAcL+RVovzWistL8jmrexPUOfQrUWEcy6qCD+G+arrRqEEaHu0tY1K2gwj80SzO4ph/6HHF
Dize0GgxRQ1ojrFAUdkaUwQfcTEKIf/K1+epXWtp8E0rKmWT0BD5ZEL3OPLMWXONJVPDb+cJUaWa
CTuFMREvPk4n9dqBNTB66P1C6/4oR/a9Rsf+9iTzzljnQxHVxP0gJod61y16owV8EiZClZ6LMDK6
KcDof+S18vC9jPoKSuCy+eoC8R0CQ84qYlnRaSehod7YN8T8sWHHPQH7xfntj71Nx/3yNyNKSxG
0NbhZjoJnCc3w99+A9184nb5NWveY3Mb9psjqPi+onh8iFtjoT61eeAUgWTCwcs7+c8SA0fdh68+
7qsH3iSGUHu6V9Z2DY3RGBqBvIQ9VJN4QbbdnFOqygeZf+AbO/4YxXvLhgZLffmJmQhagFw8GcZ
hXvW5EUMcWEUsVjd98pTz32sLzoFF8uB0cea3EpxTq+2sTxYJqwjOixOUQQRa0142233mg/3wL/U
9/V411hj1geHCLqewMqMckawAybAmXoiKkmsc0w4DjaUmK+tTFMzULTVRhghfUdecOMBR30qRkqG
ld48iRQWJ1KvamC8FFgt8HptSvujJI0rU5snSYcVe6Zg3Wlw9+3rG2s/jbizHPKWL/zml552XQ8e
UVBGpSOZQjeTprY+i/qmkgZPCJ8f9eOrfJD73g31z5JqjImNZ8mhuNT1rT79j4u9913nzz2+JOEG
u4LqejOnTtlvLEuC8H4oJiwztUShXFGfb1TnyqQvRGcx8/+wq/KZz7zGbn83T8hL7zgxTJcPSqj1
SMhAx6wI8pIlyh+WEKtiMgO8TCAIIAA6oUTSrrvMXZA0QBKyG0tyHNFLVaSWpFYtBN6+gXXQy6ec
1Z4reCQ5jfTeEayQPR2pQqltFUGGsntIGGPaoi6dHRav1WI8IWmmCgv4BDXhuvy3/1SfncF26Qt
19xFXmj5NyaqmHiKXKqMqXs7JY6okIk3vfIg7L3qRVesy5jrmPTJpO2Rm1MMdHRR90VIHnu+OssY
oW1ZKjZE0gMHJFZ82tqNT01kBMH8CeM6FnW4pinoZ70LcPJKEZet+spjbjgk4C0ERuJtYqnGj3rC
B/tItuIvMpbAkO4ppde8aNvzX7j9z5SPWeMdTgcXu0PjLxRo30hDSLQD3Hmpoihhp26TN531fvko
hd/tWysHJDD990tt91+gzy+H65+549IWIeUBQMVtdvQJhL1fS2bt7GmnJx6zYOYJ94/AnKZ3KBU
6oa9ogchM0BdlqHFDHUT5//wm3yvLP/VnbuOp1REukpRzfACaOTGV5jMjxKg4ChqgGJONZmtHJQ+
oNeqH0HynoREyRj8LcxgS0sTtK+7Lzk62Tr2bu5QwbGv3Z0f3Au25h+FSuHZX7zyVGLycEIVHUGy
yaJTS6+Pg+yGgicIbKChx7eK7c+uI+/H310r1z+nvfK88/frfe8h7S8ijDFxhc0A93VvWLDPXvui
eQAYSdAnpqkjZsS6k810o594tY7J/NjTgrcr3XsES0V1ZR0iO6N1Y66vhKvXxGM1argHmfUPg5Zw
2gsOWR2uFy6cU10mh3xzQ0fb7wOVq1NFaOpNIRQsJwoTGYHNEJ3Pl2xOnMuC+Fanh++de9zwli/4

5Wv/K5xMf0qb2Aofr1pwez0cSGVNHVA3DZ0hX/1135LFucHsm1R5KUvuTSktg/L48H4II9y5jnPI
xxxGmq8pU3CmenK0cOytDAvCwt9WR1XsveJ/eHhBiNemGNTw+guSksjTC2NhwRe/Rof/JSMKgixZ
fJ9b3ijzC0msRZOe3PUDC7WlySZSzRVRI0rGt2AgmL6yxFFRSMj/BBI80oZLeYqmE2c9nWXhSC9x
DppvLYsg8UTJAufdf2JR0K9viqDTTtiulqXKr/i6CBqC1vKmzLTCwlxGj5jMw119hgq+PLZa2+WA
rVnKCMuu+1eue/HL5cr3nW5vOrbXqsHFK9j0jQKTVQ6HZtf4Wt33H47a9XSm0BOVesgkmbGK04kS
FBD9mZGLKnJVZWFprYsHjBCgXGhy53aJgLDANAUSu4BSfbG4RY4AxUVI/k+EZcjoOZWaIMJzxHI
iIziZpXqrLhOzUrwjTYK+kgqzxp96ctNFQuLYtZT19f/ixNx33o5vv/rZXZ2VZXg3heUTOGvKXu
qrBW/k8NAZVw+Hg/DX878jRdXnk0Sfl3kCoyjXX38ltayeedCL3x7B5j7WheHDBaPbvFYhYVCz0x
YN9KhgqSM7zS/McBXmN4i16YmiNgqX494QKBx//07+Uj/3JX8r73nMFA99yqsqEGOsgfexv2hxVI
dhgsnRpLkRhboRL6ojCUVhPrusqXJs3/JoPNfJg0xbjxVak7yHalCORrB49IJt3ncMGk2oRd5Yd2
8GqTTeJXXBbwIw0FvNmglElu3XOfHDyyELM5fMLDRzfk5z/46/IHv/Mh9gLECE3WVIHMC8tsiACE9
3zgvvvZBoi9003RABsJh3+eJKl1m0tEN9vs1MuuvnwNP2JCp6tNhpRZnxXL7sOTFnnayuv6Eop7vZ6
TWlv8ogS6ndpMoedy0mX1BbakfPju1rudx5/yPywCP7pDfYyt2n0+16iEBr9Jkgm5993m7ZG1Lfe
/bcG+rXg7IxmlCtgGwV25qzZ2MHlsPXGODdpNkQXRHqj4t5Pa7H5QrfvIKefD+u5lWjkmD0z56K
Px5iMbDfwWAQq51IBBPTMsNHaWHF6l3E1c7Onh905nPjyB7vk5/IaTeQz04kxVtBjnHNKrl20Jlw
yFTYfjIS1TBH+D6jeGa3Hj7Hda40wMtxlvFffjP/24vP8975anQoaIm+8wOjF5G2aVSXiN1ZA6P
87X7WGS2gH/dyNq/FrjihMpu79jOANL071bXLEpNGa9qim0Np3gYOva561NJP4vYIRw/ABAJmze
8w+m7SkAP2tpQJ+Hs9YBe6sSw3HYOAUYt0wktfxuftncSfhRq51LwNK/t9+/3fT49pYv/0bvj4LH
vEDUHooO3KaHBOQnaGP3lXjOYowQevBXIiIhuPEoqP5hRNCvXpIrrn2dtlz98OS97bIj127KE06C
jXnU3v3y333yvjELFRG28JuTGihQMUEgTvKx1M+aXtYWKapOvE49wRSJgHHntCLv/xd8kn/uzPQ
4q6SIQUonttTZ/KRgQ8tGwAlSbAXcWDkNuhcugb3m/70Re0HwBJCMQXq61Ou+Abmc4pOtAk9cN7I
vWvrIZjJKJ6S6rObwv0t/JeCy33n6fHF0ba0Or0aXNOBYAP+If1WUq19y0R97+Iz8sf/8/P6Wjk
rQ2eKF2q++942YZV03sGEfQiUcqJyak6Uwa3BvMMTIDBYSSgMJwYAASi6sZU03hNvM3IZuoTG7Ve
xmentLIOeZRQ6USYuIrPZrYZXZD00CcMBvQDrfMADJaTqzVsibZGpleks6ANhzrbOSB+fxVtfnju
mYNnv5Nk6o+VxsLxF8q3KsON6bM6FkdXx5vLMHx5oJxjYInpmctls6GFAjLkMbL63I4/AakD80HP
zjbt59C7whNYW24VCRB4yETi4sDV2k0TRXXxr9oa99aJ+Fns8Y6lujyrk/kw3/wh9TQ/f63/CgPC
Q6210uoW4EN1jSqXVFBMgJHFD7qqDluAkFg29kX8BAnVDtUAWY4IkZPT3fL0sYQ1TWYah/jh/fs6
IF7S0JBSKEPHz4ot9z1QBzwR6CB1WOZRXnc0wOHV+UXP/jLsue22+XH3nWFLICaZ3LiX3zwUSPW1
9wCkHS6vHp9FTvLbUMmPK/+HNNcQEIHg0Gs/dgswggL+2MdgN8kfHa9TCOXc5l1/w9ExXXOPQ100
I+GqiJw3NNjCDBGyrqJkMqynvKZJnG7uzewDJTdtJ8/7YoQNLqSRLnK2hDEs+slPf/svZDtbaGm5
7iMrD/93vdk4UO+H1EINR9V6esibi6jAkA9iyZBvYpadH5uTh9eoSBvztyCh52ELHZ+YyFaSmBvA
NLUWzcZad+BAwfk6NGjbDpBvhShT9MYjXSZTgSUjlUTWci5eVo3tDIYUGUbwR2sAfmSj370T+TWG
77ALqLXRGLNpTJpjqGCVTzkPpOEs/D6bQeACj0F7Yt1Vx0/mlrql3QgieRylpXtuGki6ssxvRXJ7
iOykK6/6U5ZWd1oGSgdp5EYsshXYgDc0YRo/Vef+nu58l0/KfsP7ONMGNf+2KMPU0CuMf2J2lFOl
nInpmOMLlKrk1cV/5WVNQNu5LGJxQ11cNaoulGzkwpXshGH/3IaalsrkuMb/oRT1tfpzTiK+KfpO
7fRsIx1qn+N9WvdmPhbGiOoj5A8M8iyufilpNMx8N5A1dS9IwcP7jpu0+C1tbU3hQ99Lg4X0tKKH
17TnLLsYFDjnl9t3+NYLi4u6N5Q0xLizc9Tinyth3QP0pdYe1GFaLu2MQo10pR/NlQx0BUQXWhZJ
D1Hg0q1+aiVjaY7NiJhimlNkzppebDrwVl86i/+Qg+8SXV2my7laCmaJA9zBy9L52ANoZN3X6pLp
7z/5ARsu64WtFASDOERp3kaKVWJAcAsQx1w394n5YY771XAPRyfKOger+Pvo3WrzW3jfc1Yn//EW
98qt918bXhWjTz00ENESdX2GmJsFzqwpDUQdHRZP/ZyGhcfM/w0F6nQgyPERJc12F7acN//Z6m3
07oartfyFzwWsRA94SZk9eexEdbXe0bBirrRwAgktnXCiOGKDBQTWOTSt+jbSqRRFDYImo4kqy9T
5GxVSj+/PCBg6cel8b6tje+Idv3+ONXKqKLSAR2ALHxDB7WaxOvF2IUqPShoC+Ih8tmjyVQN7UBu
x0rU5KATiBabCCFw5Jr/dX9lXDPy6xxOXqlawQehavY5u+kpOFZXn/DtbJ8aH+km3mrhRG0Mmidp
YmtglCYG5g/nAofrfVqEgEFbSRwrKx69opjiBIk9WbaRhe7tMpmkZPhkFH18zfcTMleRfCo6qED3
10qxdNARx4pFK/Hz/bEoVW58sqfkb/4+CfkgYcelwL3F+OWRuVTvda4remTzqwX1xScM3WYTfGy
mtKrZsYzt1ZN9z5avKsxPbaqkyCJ4LzQQownyJN7ptEqjkAa4h5Bzg+y1T1lpmFlc1Mt7dqlJbnZ
83PHUZiPstlSm29hcI7KR3QB74fAkTvuKxZn3z8sTdnp+ULiC7BoJ6H10S46NFKDuLnkDKFnlVK
K3wY0cDBboiWka6OrBCIK3pWEmbdsdKaw003EzNd1lRBWlYlFgtNE40piIN7XN0oicm4GdlaIji
qTKIsCA8DrJxOnbeTYv9aSITJWY+uJ9JuumeqizQ35GDBTSPzcFE7XnghQZizkUzYVXLf4s6k
ZeLJlZR6kYAp6/VdYvng/MbjxhVH374Ybn/4cfl+7VDF9OZVM9npIgYcIIPz7Q6X3Z1xOG4kt/5L
x/hoU8ZWRtb6mxjI4NDcB5r6TQiIQxVMchZTH8ZtcqK8E/NpHSumlqnV390JUirTqccPSnhHGMwY
J45s7b7i3+f2mtr4yqJTr4wWCKlgCpVHVHAIg/eaz9DWVoDiuoUGadtng3EURBBcmkcSqFmDZ9h5
3FnRk//1ldlK+sbVyJ9cH2gXHCAs3Ez5ewuZW2aI7JkuUU91fJBqjMdVRHkz7THxixotEXCMAW40
pi68ucaNGr67IoiYlXSImPUyGs+OE2ttQsMkYqqlhrRUUkejblpPMEou05X7j7w1b/12SoqIq/U
saY3o2HfNhQLYAqH+rjOk2tgaS5z66LvpbGic+hN6W0c8rGuq7swMZSodRVjN2uZ0OVJNbwo/G6r
K4sy+euu4XKgYopTtssAdeZK6mbJmyHVSHrdm1N5J4YeiiJiCvtI9RxxMZ9sOEn+5gd1xA6yzRyT

nSm2rNGm88zveRhVKtUNwlb/YDv7Y5+/DemBhBh89TXv8/nlLV6Tf4LkVHF1mvTemp8LhrfV0H5w
5iAamah6TUcBK6bDSzrT6hzahQJlcyMqDa0O2OdjqZvCjXHC6DwMC6mHMF10VMAFdjgGZklWgdOw
o2aJ6IpIehacZ5i+FBNi5Vx0dOWuzV0akMgqeKCGiF4naosMTFIInpKjlFHRxi6zmoaq5teMhHjPS
QeapsSAXqIsEY2pFR3txRdewFQbndfB0maxrceMFk04EKSNOYCb2AVG9EKFTbhtu2w940KjkIFvm
7QUwaajouglgqs+dqRWlLAr4o7L4Bw2jq7K/fffL488sv+jcWXxr/FassfX8ToM0d5Aeor5JeVMI
2VSKX2P0a6srByRqEAoJqOSO0w0U5Dw+tf13kvWk53qg88wGqq7m0p6fd8r1BA8USveMTieIfW3Y
PBiMjPqECvuNcqMmK6GI+wwK/Df72Ea+xygCHR01D5CEvsJHDFZqt7FAYt1kJGN8TlABrIfR1NIp
9aOK2N9+lt+IAtpyHtJAgfDYrzBfSipUbHycPNKahhVWhONSsE2lYyy1OH7c1jcWxk4Ivwm106rP
Ka/SW1euCLEMx5glDhEvowLHQWkHoUy0yNSzVtGlbRvCn5V7KrSO9tAnDSCRuIMVSwi4bld/KKvY
oSfPatEzoQInHBwJmNGfygoZKUOQBBB6srS6FCj7TjvRSbtkhi6qZqVJ0XGQLqgRwUd8EsHZROH9
6FM2FhfZVS95qY7eGC1k5m2kp7Gi2WEypQuWHs9C8OF0gauWRRoEaOc3wefZNGJlTxiCRX+oVKh+
sZAFRHnnQ84ivE6r5VKgaKjzU6BOEqVSAeGDyIYyhjco2mhuGfLLKWdFxsxsberrNtlb3uImosLiex
HbMyo1mjfFb3o9jH98m0BU3byl3iYrAhXvBMOZw0bb7HL731HFlrMVo+trwx+7KWv5oJq2trcr84
hLHK9QkYiMoYZo4aaaSFQkjAOwODx2dQAD5a9vajUNN1fq0ZxxR1EsaeevaVyQ2No808njtHliXH
uOFsqzR2hZGGzu5ipJq6ibiT7WebDgTTD0qh//OPH2n7Dj1ZBprvz9okTxQI9xQ9X/ODXptfkMEL
kkru16Rk15wIUH/yCxCqK4c8PXUCN65VO+dqqImPTNxgARLAGAyZ5o2hPPDgI/LeK0e0xky0RheLQ
pGonWWmBoiyA0gng3N6A6UuZ0YgmpFo1E/MCLQGvnhmEyyvXNDyroUav3KRFjRmD1ZwkJFSWijqQI
a1jhGvioFMV/wF8iOobVr4kNud0Q3XkNbsYyxQXcWv0HCAy6/hIzFAVteS1tNemeL3c0Wfx/6edd
LlFSZfVl6UHjytj3Riuv1db7OEmpZqKjKYT8k9Tazo0iQ7Eq0IBCpNws+cTTaFy83aVAdVz0xLOL
YKpZGbkZlNiY4SebQDXTd+Z0tDCAwCp6hLjvY1nctMuSGzw6n90MrV9g2xguiPqE/lpSaxtcSNX
HLRxfRaoXxgspw6P0+4oa5Hwjo+n2oRp5pMEHFRCz+/5dQX6Gdri9RoqCpMrUAMn0uyH+R0M1uTw
Z8rp4QUIqp+/gvXx06rd2kzi6jsA6RpNFTpdL55TxJlFqaTpg0z637ywsUghBZitaQ0o9ChqbTCe
9rP8nZXtTiNxnw8hkFFDuNKZNpZ68ZnlBoJA+SgthU6tjsyj2B0jP+GptcnS6HirmsXQU1PIID4cU
Md+L5IIXGzNO3JNpxPuBlaa26Y0vrA34Rzp5plgnJnwHsvHjbH+8H/43leEWu6lupgoiwrSRMJr+
kjjy+VA/jWHAeyJOUY8AgoYoWZQTrmM4dqboKQ4eBnHCEMTSJfKXQlN3jjPypkcFdz1cNQ2a6Z7r2
FpnWPeXJmT5VAY0SBmB1Zu6hq5768q88cUX7TYthNRWWphsC65zvMYfTnOtu7PaIHOJzlaXduy0r
eQmoJ0lNkutbLFyldaGQNWASNB0xlqVdrIdADFeXZU9d94re/cfjoewMgpbVOePkivNDIzOsdil9
Q28aVNHB6Md50SSDn/VOScidDU8N+XRTQECk0Xb7WgkG1wIBNLbRWV1Na0NMJMYvMK0RLpb5Y1k
enSWHcaDSfM1kFtpNFXytf1+sta5+hzC/NcU4LPC5yxG1gmLR46lVlYZJe72iUgeJ2LUVJmUR2BB
+uXwo+thZfGurG6diVSWALnIS2Z23gFKnZNIX2xRU9VxpRYGyAqZwIM8FLwupPp6OkvbLtaA0tz5
BjUkxqsNoG0G2rcVRPBooYwamhSo2prCCURjR4bSjZK7fMxHBniAm+YHcv4LL4iIIip/26ZzAhDQH
EG9ihWOkJsBiTtV0a8sHM6lk05rxclqayzFxlG7OoSyMpWpH86AILSWKqlTvE6U1j9fd2OILJmly
arWV5rMZ1d/NzVVCq9lOzzNmMrTgTTJlZRsrl9ZEhJZ+8iR/TQ6B9srXlDaavCkv4VitCM7qa39V
Ok/PC/OGJyrmuc96x3YZvakMgogiPWVSy5WPFuoQYhk4GcC2R0ACT1h8JswcZILfCmu/Sr6ZybW
dCMNv9U3CAzzDozviQdhdechHBegiDe+9jsuCUZ6Ge4Dd5+YNAgK+saIy3iAmGc6TG1q3U4dqkOWo
zYQvA7PMxtB9NIOra1uB9bdEYHYaydWY3q64+CI1EHesN4q7+yuGiqnnlgXXA4Htjzzz1DBGvz1
P1IjbkRwfr4WSy2yueMXWmskES/C3PetOPUzpxURz9iWYRHVO/m01A78LS6Nk9K4qOHw3W57fa75
NDKGutNprxZEoEfXbUGJ4V75u0pX/fQRkikIax8XBHxxKIG5jDC9bUNyAbT4HRc06N6RZxTwtAKr
duTqiUvpB3YJe49BAFG5Vg1kNMOWbx2xYuKf0f9CzUPRHu9ZrLZNNggRvW+/Ma2R2c311f2VLds
jjOaQXMq917QNJAEnmxDoaIoIssK//zx/6oPi6MdW119cr10TAYaqEHmZlqxcYRONLcWh5S4Kyfc
ZiOCOKqEXqz1NCmTeX07HZh1XlInOQ+AOBp30AR3nmsSH9C80k9YR7naYaoNf6qmNjH6MGMIM5r
VtkTiXtoqTdLzwnNijc4COWXOFEBDxz2O+Ln7I0HoRNO89QwLnpGTvw24mlorgfRHnhz0qV/V1jO
LGfQVQt1tdl5egR+cLNDxIW2CSldXZL/XxRC3jW4GobUWiyiL/ta7rFv114HV3kl1cc2pNK/z/5
VCrcvSrt19zEEbsVjuYI1Wce27fx2gtMaBDOZrIwmBe5kBusI5t1XFckCmF4x9PJqxp4ZDm5xdZL
kHLGZ+HHeWymklvnTHjxtjlsHbT39gBtnozRo8udW0puOOJ7WdG/5b28v8tDX7HD77p3PXhxdT8
NkOMregmEclv7GsWcf0JsGL9jIjoIcbNw2pTa+ReVKrpnH9X2OND6R1rj0bwdvcLp6q4FajoxHp8
j7tKuhtdYWpRggcEG4Tz+IDBwqJQAHPGLFFKzYtErn4gudHkjYiQZLMkt171C44JlHkKgAaSL96i
/OycOKp7WIog/1VcaMc3YnqCXNloo1sbGTkAphyolH11lsRVVtieaKMaTbGnOWSdiRvVb4JUs1qW
PYCuPcnb6NwSUGh8dWT4o10FJtdHjiqQO2NEzhk91sBCAYABrGG+OZ5VQZ77mw4afprhLTKyNsu
EQM78VUA/iWEXhGs47/MDNXKfj3bvVZPh0XeobOzk6DN1HBMz5HZMk3UVJ7tLWqZJFTHZjC0Z
VR4rMnfHBFgOhw0311VdTaaJInnbGdzNr+CwnyuGklIexbyeTlM4bYwNS7V4yrIv9FhdKPr6lNDm
+hDMQEsLq4qOjBB85gY1tcaWWqyd1Ld4JlqrM5qPdJR20rT6vLYw/GvIernlqxsWexzmzqGuzXXV
/TaJg6ziJRMEXjmpFB+KuFx3EAnsnnX2UxCp3RAy2YZNBTS2OKMxVT4cYWKVUm64LDPR5tsAN87
U23WAQx6l9jEp1mIdYlQIzPHKUhgRTBFI39WtiI8pJ6l3cLd0OZ6qZrIb0dzweyXwP4t2p6SK3h
92VJyDH468PcDHev+qF+2LEcaNO0vCk5qjHtVsSrmvkIvFGGYBzSdjJfG17BkhXSV2HN31RGGTf
j26TT2ZqVOjZKq17v73eK2iTqWfaNr6jl5a5L30j5/1xvrW7/++LeHGvxmfEjjW0ob+kd1l1Dhge
/GASsLkpppaie4KrZzLWmc22qi0prSdJTiIaaIHuvJGECJrJTYTtNV/RgCw6ouRiXCRA8LV6qg/0

tXk8bleBKY3EqMovrb7vHNCbTSgkRHSaCgrjeY6PpLEIXnSQSRpCrjplDMZLbLK1j/Yz/MT1k5e1
y5oTYjh1EZCugakpMLChEuobgpR9cjKBp1J+EDam+ZCqTqyaIx5F5UbirkMEGOxFR9+jVWqWQizq
z668W13NIiUTUBE2+X1Za0/8zTySx3q6WssIm7bmjclmZAKgGU38tmVAMxqyYSqbBRFVhUjLgVX
zPvLYT6NOFYyyafEagRAROV7cyRyihTcky9mmiTr5fE9L5L3/Mxjn6Wirxmr9kT4/6GbGAY/s2Dz
3pjXT+68rbRaLQETwWPh9WH8IA9Y0k0nJ8GQyot/QnebZidRCQJ66hSU7iJTKMMZWokbYp3YbVGq
g9L9XOTqCukRldHr9kqF6QxpYWhsFvpDJlUubT+/6t0BiJTiaBr+7zo/PMZOWuT4JxJHTs1b+PKj
JEdoyiphZPOsLpUlfygWUz8LtknjYqr1UYO9ENmKCDtnan+00aoV6+76TaFYsZ9pa/hfbiDtlAyv
NfinYPoUqOJ9QG88eSUQfxcAdBA7amg2IKoJJYx+w8dDg6ipvGmeRbxtT0uYJ7tqMbUONUNfN54A
kEf7YTh+igYam7XqqwcrNacTQuL8LrEcj/jKh50LLhGz4BqxZUTmw0jLdIWYzSlSdvrUqVJMqtq8
TSDFnf8orP98LpzMuC5Qb0aHM3wWW2sV737nd14PH47xjW4KZh1rY4nqv0rAAAno+kVazErWKqYd2
9RxcxzGEQtz8/TiM905p0+JRTKrxVw93W9um8K0cDTdwKZzTAX3J7o3NZnt8npqRNU+7jhRUGKdG
MoJ8LdwVy960SV6OI0EECVavG5KJY6JMFZQQLiBwkGU37xd03A0oIxY3RhWWWesen3czWN7e6iRX
Onu2jpeM9FqMNTTrbpGDR1dJto8Dfhx4NFiyjoSmI3TKsuXl+jY3ywjgoPJOFMQa11BBQiAlzubYw
Hdd/YH3OjxDJEs4tqkzfnZC9Jq6ZTpV/q4qJ+sGn6UthJBkb3JJJxHthEsEoGEBDSR2pes41oPsD
IazRWm43fAaa9hLVCsirnJ9pbytrVtlibaByZlrkkbVw+5Z6BETrn+fSwz8kgGVlq394oc+0jyrj
fXAk0++JtzkszFXpaqc1RTciM19yC20jA801RSEclh4aXaJK6KWOLLhHk14eEWtNKlED15bupjF9
DRV7ic8veM7PT9Kswj2TkzBUGzREeCOiPaUcUlaVTWkQZmnj9vHwvM6dcdW2XXaLlEKrkp199k9K
WIG6suRK/1NJfMdououbT7VOqNMOzUhnYeD0Dm20cXP8XUbreGAvz16ZFn+5fqB4wiC1+JR3Oo3N
dTc1Os7js74w10WS6qoByXSWyNPEu+Ga6ZBcP7cQJ586imTjRU21noOiI+c3jZST7lQWnHA6RTqk
ylJ5r1M4YbT8dSUHyqqb4xGE4MjppSchYPHtoKpAWvEqBKcBIoK2QEDz11j4TPP9fvWSW4icV3Ra
5kuq/JN6EnSue91ZPKwGZfZGko7aKrGiAyKx18ER/Pz/9a282U31rX19bcV1uFErYHDB0QLDLAoR
twCV5MZkRMcq3Wq8iQh34JVCrllLVta67cw5p03aRExt4dWgtN00yn3ZkLlKxODxTdOieOAZyYEV3
RsJrKtW8Zk2p5paVfKpzJVatG50iRM+z+7zzqP8ibiMSWdHigMM2AzKM9tPA5DiRDeuhXuzuP2Uz
hg1mwHk43MUVoZSgQ0GsVTplYxrKBFVJ5NVufn20+ToxshKhCam2z4fbXz+jDl3U8VuNmh0dvNmQ
BH4uR63A6jOLzezJRK76ixlQjRcW9sIBjW2iKRFs21VImfjTSersSaZrT0gBRCNKBDILTn14wdNI
4x2MDedUuw95cY5VeJqZ1aTmBCuPfiaqS1UdibRQshWkk56rUffOMjW6FIHaXlmgYw6w0Yi/8fs6
lpFDXj99oy02cb3WAKp+aeF1cb6jJf/wNnFdHpZV+kBAHOkRkCUwEntQHYS4tngl1VRqrLgBrFcLF
QNDpEQneFqOVQYyaUhUZ8SsK6ri++oK32dacLeLIZHSVvOXwt0dcH3ZJHJmELTOssHaGm3ysPmEE
qrxua0uFs5Rr168m/NEp6aBaasJfMxYROMjXnZOqtFqnBPiPRZPPi16G6bWJMa3OF5dpQiE0STq/
fAgmcwIFAvXltfkmhtvaQ1Vsrh7xhtKSURiVzrSWocXTRLum2lsV2pH3oa1rxl1Y1vsSIHrabad00
BnLwcOHOla04fxZDrBetAsrnYjNe1MbNBf1aubcVJVqGfQGMqlLRlQYaj944wFWWZLZFppLVREbV
ojAJHVAWrSnNWyrQuHpdGZzlobLoUtrpuH5FXaPuivMu2R8ovPRhMy6zbK2QZqKdYbbdly4tFnt
bGura6+rR5NMgedxw9flyrO1U9l0PRlGfLehgCSB7Mk+97nXHGNCTzfPI6JLFOURnmohPVZPQnGB
veBNjqiaawRRGND6oWuMGLnNrKoM9v1WR/TpW0MzN9+nWqLvpQIrXheeCk4ivN3X9gaZKWURBJzu
3rJpE5UHoURtZyoEfi+zxBtF7aZxhaI8FiTWNcRvK8mX4mqm5caoXlmyfmgAvZvuOk2ObS8artmk
jjjVaV71FkKeq8MZidxsXBHFBxkCFG2kxMIiKhc0mLOlZnNnVHKHDx0xNQIFYWFFBiOqxFLjORp7
+MkfxCRoGiAQDgG/Mnme1IMp4yWmFfDUCmjOqltTUDpnw/Puid4YXFxy0wdqqltGmtOckRJdzWGB
FFbxEyOEUFzaYDU7TjHF4G1HYyOvjCYDNbe83NXN//W9vNlQzD92i9dnU3Wh2+eUGiqJJKGyBNRA
q8+TDU01BNws6puWHKnjIO0iWjhwiCtzTiKsejiY5XcIF+Zd05XR8NWkUlGA0NUsY5m3HjdaPolh
S1KJD63rAyZVcOzv59/9mmytGWzws+SdikT0lN2MquOo8rIx6KndjJf/Ak7w5fn7fXacY4bqqezn
InaCovEBvdoDoF+hj21/3LtdRZJ2mvz1K9KdJ2kGBE/abefaURNlEHdTM6aL6q00Yq09TPqGyHl4
GwPr6xa9lCHZugvWLiOiJge0/WlWlHJ7bpzFcharSdzcnhVqbDPZtVGCHZsVGG3bqWafxgyF5ZZx
ES2Nb+0yJS3m7qyLicSTBt6pY26UmMXRayvQS+zelavKc62PVXOWseDZ5A2Ldc16+XDpf1NP/1M2
NCXLbLef889rw43eCd5lwZA963T7Z5VVWEg9G5OW+eAj5WmCxuNNTWURbtUXGAbgQktJlONN/GUz
WhuhtpBQwWNGcwisaAKUaaxHSlVeIhJqlxVHChVWzd9HWreVjPwMxgzOaD28M4//wUGgseByHrLI
9eYVqxtpCvalVmALEproJUhuq5sNVOobJGQuB2je20opqoworkibxqSzwskh910/Q1ydhVi9aaRy
TMtCRPOdjNLqcuompCLS+jIzD4XJeJonZpZLZ9ZzqCb1XU7/MbaOmtK3pvEI1pm+N52419ihHbci
pz3V8fcwVPfIKhMNV9wuI2muCoj4xWXYuFfiG+ariBteXJzn992RqnREkkCyrBqIvEisWXPsbML6
VsfXfC2ny2K/B27NjNRz2sYwlnHjZJcGHT/N89q411uL7x5onJZygeXVXW2dGtrAtI3R2tAUEOC
7XpXf9hZtPJNKY0vX50+CB3CdVaP0pn+VH04uF7vSyzNDA15QjdAePgee8ItvqwhsAJaV1h4IzED
o8P/2dmbmkSGSOLc4lcdNFFjKpqKBVFtXHAi1CLa5PMZFjY5NFuLKUvS05+ZfOu01UbqNF6m13Id
rpLOl/Jzm7dwgRxLdgoEO7R4UPL8s9fuJXaSlx7aF5fkm4K2qpCJfXlORDuyCzFWUhmVlK1j1IrQ
dJOrYmoh+sbjYaysbFhavW6PYGJbKo4XAcW4PX6wsIOqmHBedBxpyppwR8dj162Eeq3FQWlD4wuNs
L60ZRHqUTSa+tancKplVHZItCZltEe54cqEuK+TJJIXSkAhHULYM3J5J909dmTThS7q5EKnfR25/
vJVv//y8FlrrL/327+1NNzYeC29F4w0aaOLjtFUYGHMmRGcY+qGNRF5L/Id8cBAH5uMJ7bqwbS6b
PaZpG0t4ewZUuVqRTAlvh+Hc1KteWuTYODms1QbTFU9MZxwGmU8faFyfd02YapYy51y4mY57cyzI
jE889YVZDuL0tTbW9QPjJTKCEURqW6LO87STCNT8ndC8nul4AjvdtA6xT2SwwHFAxc0pMBIf5fX1

```
oiS6q5SrIjMMgdgxifcEs7ZlsEXNctx6VBXKEzdYDvMnLSjewVD1WhlIw8DnKTGvDmWmdRN54m1L
mtx8QHry3H0xJHeaMzOMBxRzfNgyH7Ohe8BBKGcVjXwsjBObeIj9so0nMuIX2HXtpe3kMhEm5fMo
KxT3DVCb1J3v9Z11hGBpYSMop9nf/1M2dGXxVhvu/6m105H43ndp1p36gGkV6VG0loPPY230Fii+
1hUbQBdtqanuRg9YVPG/ZmNM2SQImepedmKXlpb7TaSqKpIicvSVopUu7vtuj6xJVGnqUGktQuuN
db9tcgCnHKiRovE6bzzzpV+iAC1qohFIEbONRwGa7OU1ZrPwTGVNESAGQbbTyZhQZk1MO6c2GGFs
2lUZblQ2kpH0ZSWQujBGRxdXpMvhBQYnwt1qX/uRIHlOt7CtffycB9EA6jguovGulwbwyA6RQyde
c09WgFMkCoXWA3VYI98kLrdD/NGdIglk47sSiuu3e23uuayLwHDD6dFGRd14bmB5+qia4iW6PKqC
qbs40rb4wpHyiylUlBz04xjNzjCHVodMfcMzMAGVnNmZFa75AlfUtVl6Tgbqbaz2Ye+c95bCff56
melsa6urbwGzaLKdHuYkjk7A2ss6iJKhJAiFZdNqcHyQeKmis7WOP+ydj+6vb5BG4cE+1I0tVVD5
JawWiF93BJWK+A/SR3P2rgIX8SoeioMY+vrFkKDSTQzNLLK2DvY+Iax007dL1TZj6LseF89HJzfH
s+ZuxQp9W4ratROJ6OQtK7lhK3bI7uFURBSJaVG+8brei6/UgCFmEfHQZ4WI7nx5ptkZTjVDm2HT
J64flPnunlgxQyTKWqbwrb14XVnc3obWTJrSm2sb6jEiTXoEpNQwSspoKEi6uhLcYkjCwYghFojt
RpnzfqdpIhSMxPqQVNrWCz1TaJDnowrOgWimeoyquLj+zk3zgli9PQg4Wky6mXIy/g8Ff/GBby9g
xwXb9XVDILN56pp5O1SyWJtcfOm5Wetsb7+m181Hw7lZUWlCHrdw6JemQyXpLB6wJEijaoaQgups
d0FMDinwXGRb9mheSXWfcxYG2EjWUTVuk0iDq8xbqgrG6JXim6hKr/WhFExQLcWU4+DMTTXEU1la
W3lLqYz1sh7bh705Oxznx+hZ013glplbCKk7gsLBm4Q28421U5w+D0IxqpjIGF0TNBA8ihkKCnCA
d3pGYRwCn2lEOH+9dqBVFa0k55FppA3Q+z1WdeL3pd2x2hqCC3NJW3FlDo8CncnEYGEearSE704Z
Z1FS6YCa7moEkSXNugqCo2hpYgNr3RVxVw2b4uegsFMR4z8UAqB8aQGu1TCem7Iq9K2oYc/x3qvw
Vl1wL2Pitj1TyUKwXVHNN11HU1Hff9YHHDROsKeu0BZu+nYtpcUvbn5T3zgV369eaZs6Rkf3YRD9
JLxdHpiZyglLlXyFjpS2TqLqakYP7St3nU4bxDVeMNG0OmJcDc1WHSQNY8tmSSMz9yU25nVs3hgB
35nks4M610QjVlei8xlrUJrrD8zR+/XM5KUSObPOMWHbN95Slys2+Z4tl4xHD7U1N1Glmv5YjaKg
zLYchLHVZVDDxvl57ZZiGYRtYP+bfRQjQu55ebb5clDR57O1+ymoB0YHdUX4YCaVsIl656Gur0n8
aCawePwApureZEpDsJgjdeZd+pBb8B1GzKZYbQ4aT/xlSTWOBqNlduKLjCj4lw/rm7kjLeeksOKO
laYbS3I1q1bYzc4z1KbJkqNOplTbj6ZttDCWSZNZ4ds1s78/LE4cXbOheYglZCa/8IzaUvPeGQNH
+ilo+G64ih9ZiXcLKQymDyUJpAlehC8kcFqNbHRhzRxxYPOvBqrZxNCzxBRVUV+SjU8NYqyJRTSC
j3r9xJd7VjrGkTO1jId0GdikT+VqGbIzLFu4szFpUuQEVBcv7v0xRcQOVUW0zaNTC2NR/YAGMnk3
FLJkAKjy102KhIW3m+weYsCWhFVbZ9qTa5vWudFGVWp2ntLLHXBFPiz//Q5JaJ3DCuVvPaErB8HI
nQU/TG7LKYq29rdkp4kx+yFSbQOxXOcsZx1ylwWT/WMC9yyp9bq5an6RR38n35tBh+za3NDEoo
hoF+KXTsogQTaCZMmvy+flkoqhlvj/PmSpq16fhrm0c2BImnDlTKa3QzhLFwhsHnHQICqjp62bGy
XQZwsc6HziyuSxf/pXf/v3VZ9KWnvHIOlcvBwEclfar0wriAZpagK1EZJpxEiVUati5UXTij+jE
zqXqYRDITroce0gAL8HIQViQ8EN0VZGRHrXjMe0jrcPGkJoitGBI33coIioqXTjaNJ9YJlYhem
5bmZfeFF7cdY2t8JV4oVvoZUJv65jlQ8HDQi/AlY18XNqmoenNNKu+iSZXjunmYdBoTALFr7XEowv
+vOPfL4U4dmUjTppuKouTrspC4wwbZTxLFU4zW9dLqiPq8Ml+UzT/2stfQa2xGUKr6acAmTA4V0q
HNiXWBN0tBF1vFW0tQ2d58W47aGBEAmbYXhcJ94pgYLSrBpiVzXbg06A2JIZnsQ/llYYxsZqKraT
n53juqqD12BuFj/J8c4NXzwNC/y3txPPd029IxHlOWFpUsG82uytjamxAg4mdT59UXBidZvmvL5H
FL5qI4ASm38AQ8LVk5tC5Y4KA8PbMkWCs+kKK12kbM0t46uRFnM2udsMURWLTGZkf0dCSUdhoh3M
rstfLF9Ouc/7yzZvuPkjgq+1UL+poancGHphDVXqFWHazIBnjcY7pZt5+i272DASY1VldoQmawfD
hF7nhrKXnvWZCalbDRhdeM//MO12kVptcvbYp+bp/MvxRylobo407VGk3ZKJapUJUK6M6oA+yUyf
xpfwKSZEI4Sxk2MgqIOMbRbU2OySIQs6lLjJpLnCwqx615bLEIuo1xskmmvgM4y95EuxVTXWVXdD
W9eH9e+Xd3mz00iMxEy/lmp1lTemct2Db+LV0oafPdrWJsyl/cO9frZJ59pW3rGI2sxHJ+6NL+gW
617yqqZDSdx1QMXGosq0pX17LDZxZ81vVmlDbdvrnQejI/N5gRMosDb8fbJqoL5J62tu6y73D1j
eMu5xFhd/Z8Shsp+GihSzqGM0En+NJLLw3GNODD99EA436tIm+sTdm8bUcCADZg7Wix1RoVeGDUq
9PJmuRz87K+fFD2PXC9r90cXZeXAYzoPrV0mVLuk6ALfe89dcs/DD+nnsUlnGjG149x0D5kZqkeM+
FlrbQ410gqlld190qnOwOMKxk5nlwZiCCexDrcDDdqmXdvMkQ7/s7ZGWWO8XPxbOATfKseF2uhIh
3MTNd43B/prA4c8sVqfdMuqisqCXSkW/z3br8gMO14/zQhnz189Kz/abRy2GQoRXXf+pw9/dPqsN
tY3ftfr+sPxSLVz8gGXTUGVHg+flfJSWSVV1RyTwhhYn9uym2gc2JIdCctZxnoFy5J9hQFurjat9
O95kn6JZcLyJeuOmcZW220Jf854ZG+thAh5wuK8nHfB+cZnbNUCGzuknlZqaLbInWmTpbGxBA7lw
rYd4aCuS96blw7o3ILMbT6BQP7x2jKWALWHOxE7pBP5zD9+Xue3EQndNkrIlqlq05dqItjftb19
R+pGiScpu/q8ZlnZpKdvM70/tp4X7P2nsJYdE9MHjV4u4bAn8nTCDaITZ3URCoKjmx6HcqZ7rHJ4
1b4wlg2cc2HY3JdID7T63C9JOecdq8hmUFh5Xwuvmy5mzk5Eq6rGdxtLplDWale5K1fjhHo/xJgA
HavyI3ijlo8AAAAAE1FTkSuQmCC',
    height: 150, width: 180, top: 290, left: 50}}],
    { index: 2, value: 'Gender: ' },
    { index: 3, value: ': Female' }
  ],
  {
```



```

        index: 13,
        cells: [
            { index: 2, value: 'Contact Preference' },
            { index: 3, value: ': Phone' }
        ]
    },
    {
        index: 14,
        cells: [
            { index: 2, value: 'Email' },
            { index: 3, value: ': mary@gmail.com' }
        ]
    },
    {
        index: 15,
        cells: [
            { index: 2, value: 'Date of Birth' },
            { index: 3, value: ': Dec 8, 1989' }
        ]
    },
    {
        index: 16,
        cells: [
            { index: 2, value: 'Department' },
            { index: 3, value: ': HR' }
        ]
    },
    {
        index: 17,
        height: 40,
        cells: [
            { index: 2, value: 'IsActive', style: { verticalAlign:
'top' } },
            { index: 3, value: ': True', style: { verticalAlign:
'top' } }
        ]
    }
],
columns: [{ width: 20 }, { width: 280 }, { width: 172 }, { width:
160 }],
selectedRange: 'B2'
}
];
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheet,
    showFormulaBar: false,
    showRibbon: false,
    created: (): void => {
        spreadsheet.merge('B2:D2');
        spreadsheet.merge('B11:D11');
        spreadsheet.cellFormat({ fontWeight: 'bold', verticalAlign:
'middle', backgroundColor: '#1167b1', color: '#ffffff' }, 'B2');
        spreadsheet.cellFormat({ fontWeight: 'bold', verticalAlign:
'middle', backgroundColor: '#1167b1', color: '#ffffff' }, 'B11');
        spreadsheet.cellFormat({ fontWeight: 'bold' }, 'C3:C9');
        spreadsheet.cellFormat({ fontWeight: 'bold' }, 'C12:C18');
    }
});

```

```

        spreadsheet.setBorder({ border: '1px solid #1167b1' }, 'B2:D9',
'Outer');
        spreadsheet.setBorder({ border: '1px solid #1167b1' },
'B11:D18', 'Outer');
    }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>

<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Image

The following features have some limitations in Image:

- Corner resizing option in the image element.
- Copy and paste the external image.

Chart

A chart is a graphical representation of data, that organizes and represents a set of numerical or qualitative data. It mostly displays the selected range of data in terms of x-axis and y-axis. You can use the [allowChart](#) property to enable or disable the chart functionality.

The default value for the [allowChart](#) property is `true`.

Types of chart

The following types of charts are available in the Spreadsheet.

- * Column Chart
- * Bar Chart
- * Area Chart
- * Line Chart
- * Pie Chart
- * Scatter Chart

Insert Chart

You can insert the chart by using one of the following ways,

- Select the chart icon in the Ribbon toolbar under the Insert Tab.
- Use the [insertChart\(\)](#) method programmatically.

The available parameter in the [insertChart\(\)](#) method is,

Parameter	Type	Description
chart	<code>ChartModel</code>	Specifies the options to insert a chart in the spreadsheet.

The available arguments in the `ChartModel` are:

- `type`: Specifies the type of chart.
- `theme`: Specifies the theme of a chart.
- `isSeriesInRows`: Specifies to switch the row or a column.
- `range`: Specifies the selected range or specified range.

- **id:** Specifies the chart element id.
- **markerSettings:** Specifies the marker settings. The marker is used to provide information about the data points in the series and is currently only applicable to the line chart.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel, getFormatFromType }
from '@syncfusion/ej2-spreadsheet';
import { chartData } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [
    {
        name: 'Book Sales',
        ranges: [{ dataSource: chartData, startCell: 'A3' }],
        rows: [
            {
                height: 30,
                cells: [
                    {
                        value: 'Book Sales 2016-2020',
                        style: {
                            backgroundColor: '#357cd2', color:
'#fff',
                            fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle'
                        }
                    }
                ]
            }
        ]
    },
    {
        columns: [
            { width: 110 }, { width: 100 }, { width: 100 }, { width:
100 }, { width: 100 }, { width: 100 }
        ]
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: function () {
        spreadsheet.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
        spreadsheet.numberFormat(getFormatFromType('Currency'),
'B4:F8');
        spreadsheet.merge('A1:F1');
        //Render Column chart
        spreadsheet.insertChart([{type: 'Column', theme:
'Bootstrap5Dark', range: 'A3:B6', id: 'column-chart' }]);
        //Render Line chart with Marker
        spreadsheet.insertChart([{type: 'Line', range: 'A3:B6',
markerSettings: {visible: true, shape: 'Circle', isFilled: false, size: 10,
border: {width: 2, color: '#3cb371'}}, id: 'line-chart' }]);
    }
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<style>
  #line-chart_overlay {
    left: 500px !important;
  }
</style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Delete Chart

- If you want to delete the chart, just select the chart, and then press the Delete key.
- Use the [deleteChart\(\)](#) method programmatically.

The available parameter in the [deleteChart\(\)](#) method is,

Parameter	Type	Description
id	string	Specifies the id of the chart element to be deleted.

Chart Customization

Chart feature allows you to view and insert a chart in a spreadsheet, and you can change the height and width of the chart by resizing and moving it to another position.

- You can change the height and width of the chart by resizing.
- You can change the position of the chart by drag and drop.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel, RowModel, getFormatFromType }
from '@syncfusion/ej2-spreadsheet';
import { chartData } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [
  {
    name: 'Book Sales',
    ranges: [{ dataSource: chartData, startCell: 'A3' }],
    rows: [
      {
        height: 30,
        cells: [
          {
            value: 'Book Sales 2016-2020',
            style: {
              backgroundColor: '#357cd2', color:
'#fff',
fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle'
            }
          }
        ]
      }
    ]
  },
  {
    cells: [

```

```

        { index: 7, chart: [{ type: 'Column', range:
'A3:F8' }] }
    ]
    },
    ],
    columns: [
        { width: 110 }, { width: 100 }, { width: 100 }, { width:
100 }, { width: 100 }, { width: 100 }
    ]
    }
    ];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: function () {
        spreadsheet.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
        spreadsheet.numberFormat(getFormatFromType('Currency'),
'B4:F8');
        spreadsheet.merge('A1:F1');
    }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">

```

```

<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization of line chart markers

Using the [actionBegin](#) event, you can change the shape, size, fill color, and border of the line chart marker. In the following example, you can see the modified marker appearance, such as shape and size, while creating the line chart with UI interaction.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel, RowModel, getFormatFromType,
BeforeChartEventArgs } from '@syncfusion/ej2-spreadsheet';
import { chartData } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [
  {
    name: 'Book Sales',
    ranges: [{ dataSource: chartData, startCell: 'A3' }],
    rows: [
      {
        height: 30,
        cells: [
          {
            value: 'Book Sales 2016-2020',
            style: {
              backgroundColor: '#357cd2', color:
'#fff',
fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle'
            }
          }
        ]
      }
    ]
  }
]

```



```

        ],
        columns: [
            { width: 110 }, { width: 100 }, { width: 100 }, { width:
100 }, { width: 100 }, { width: 100 }
        ]
    }
};
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: function () {
        spreadsheet.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
        spreadsheet.numberFormat(getFormatFromType('Currency'),
'B4:F8');
        spreadsheet.merge('A1:F1');
    },
    actionBegin: (args: BeforeChartEventArgs) => {
        if (args.action === 'beforeInsertChart' &&
args.args.eventArgs.type.includes('Line')) {
            args.args.eventArgs.markerSettings.shape = 'Triangle';
            args.args.eventArgs.markerSettings.isFilled = false;
            args.args.eventArgs.markerSettings.size = 10;
        }
    }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Chart

The following features have some limitations in the Chart:

- Insert row/delete row between the chart data source will not reflect the chart.
- Copy/paste into the chart data source will not reflect the chart.
- Corner resizing option in chart element.

See Also

- [Formatting](#)
- [Rows and columns](#)
- [Hyperlink](#)
- [Sorting](#)

Rows and columns in EJ2 JavaScript Spreadsheet control

Spreadsheet is a tabular format consisting of rows and columns. The intersection point of rows and columns are called as cells. The list of operations that you can perform in rows and columns are,

- Insert
- Delete
- Show and Hide

Insert

You can insert rows or columns anywhere in a spreadsheet. Use the [allowInsert](#) property to enable or disable the insert option in Spreadsheet.

Row

The rows can be inserted in the following ways,

- Using [insertRow](#) method, you can insert the rows once the component is loaded.
- Using context menu, insert the empty rows in the desired position.

The following code example shows the options for inserting rows in the spreadsheet.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
 '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [
  { width: 20 }, { width: 90 }, { width: 220 }, { width: 90 }, { width:
  140 }, { width: 90 }, { width: 100 }, { width: 100 }
];
let sheets: SheetModel[] = [{ ranges: [{ dataSource: data, startCell: 'B1'
  }], columns: columns }];
// Rows model that is going to insert dynamically
let rowsModel: RowModel[] = [{
  index: 9, // Need to specify the index for the first row collection, the
  specified rows will be inserted in this index.
  cells: [{ value: '' }, { value: '8' }, { value: 'Northwoods Cranberry
  Sauce' }, { value: '3' }, { value: '12 - 12 oz jars' },
    { value: '40.00' }, { value: '6' }, { value: 'false' } ]
},
{
  cells: [{ value: '' }, { value: '9' }, { value: 'Mishi Kobe Niku' }, {
  value: '4' }, { value: '18 - 500 g pkgs.' }, { value: '97.00' }, { value:
  '29' }, { value: 'true' } ]
}];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,
  created: (): void => {
    // Applies style formatting before inserting the rows
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
    'B1:H1');
    // inserting a empty row at 0th index
    spreadsheet.insertRow();
    // inserting 2 rows at the 9th index with data
    spreadsheet.insertRow(rowsModel);
    // Applies style formatting after the rows are inserted
    spreadsheet.cellFormat({ textAlign: 'center' }, 'B3:B12');
    spreadsheet.cellFormat({ textAlign: 'center' }, 'D3:D12');
    spreadsheet.cellFormat({ textAlign: 'center' }, 'F3:H12');
  },
  // Removed the unwanted support for this samples
  showRibbon: false, showFormulaBar: false, showSheetTabs: false
});
```

```
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column

The columns can be inserted in the following ways,

- Using [insertColumn](#) method, you can insert the columns once the component is loaded.
- Using context menu, insert the empty columns in the desired position.

The following code example shows the options for inserting columns in the spreadsheet.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel, CellModel, getCellAddress }
from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 90 }, { width: 220 }, { width: 90 },
{ width: 140 }, { width: 100 }, { width: 100 }];
let sheets: SheetModel[] = [{ ranges: [{ dataSource: data, startCell: 'A2'
}], columns: columns }];
// Cells model that you are going to update in the inserted 5th column
dynamically
let cellsModel: CellModel[] = [{ value: 'Unit Price', style: { fontWeight:
'bold', textAlign: 'center' } }, { value: '18.00' },
{ value: '19.00' }, { value: '10.00' }, { value: '22.00' }, { value:
'21.35' }, { value: '25.00' }, { value: '30.00' },
{ value: '21.00' }, { value: '40.00' }, { value: '97.00' }];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,
  created: (): void => {
    // Applies style formatting before inserting the column
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A2:G2');
    // inserting a empty column at 0th index
    spreadsheet.insertColumn();
    // inserting 1 column at the 5th index with column model
    spreadsheet.insertColumn([{ index: 5, width: 90 }]);
    let rowIndex: number = 1;
    // Updating the 5th column data
    cellsModel.forEach((cell: CellModel): void => {
      spreadsheet.updateCell(cell, getCellAddress(rowIndex, 5));
      rowIndex++;
    });
    // Applies style formatting after the columns are inserted
    spreadsheet.cellFormat({ textAlign: 'center' }, 'B3:B12');
    spreadsheet.cellFormat({ textAlign: 'center' }, 'D3:D12');
    spreadsheet.cellFormat({ textAlign: 'center' }, 'F3:H12');
  },
  // Removed the unwanted support for this samples
  showRibbon: false, showFormulaBar: false, showSheetTabs: false
});

```

```
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Delete

Delete support provides an option for deleting the rows and columns in the spreadsheet. Use [allowDelete](#) property to enable or disable the delete option in Spreadsheet.

The rows and columns can be deleted dynamically in the following ways,

- Using [delete](#) method, you can delete the loaded rows and columns.
- Using context menu, you can delete the selected rows and columns.

The following code example shows the delete operation of rows and columns in the spreadsheet.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 90 }, { width: 220 }, { width: 90 }, { width: 140 }, { width: 90 }, { width: 100 }, { width: 100 }];
let sheets: SheetModel[] = [{ name: 'Sheet1', ranges: [{ dataSource: data }], columns: columns }, { name: 'Sheet2', ranges: [{ dataSource: data }], columns: columns }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    allowDelete: true, // to enable or disable the delete option in Spreadsheet
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' }, 'A1:H1');
        // deleting the rows from 8th to 10th index. To delete row, the third argument of enum type is passed as 'Row', the last argument specifies the sheet name or index in which the delete operation will perform. By default, active sheet will be considered. It is applicable only for model type Row and Column.
        spreadsheet.delete(8, 10, 'Row', 0); // startIndex, endIndex, Row, sheet index
        // deleting the 2nd and 5th indexed columns
        spreadsheet.delete(2, 2, 'Column', 'Sheet2');
        spreadsheet.delete(5, 5, 'Column');
        spreadsheet.delete(0, 0, "Sheet"); // delete the first sheet. sheet index starts from 0
        // Applies style formatting after deleted the rows and columns
        spreadsheet.cellFormat({ textAlign: 'center' }, 'A2:A8');
        spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:G8');
    },
    // Removed the unwanted support for this samples
    showRibbon: false, showFormulaBar: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```



```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of insert and delete

The following features have some limitations in Insert/Delete:

- Insert row/column between the formatting applied cells.
- Insert row/column between the data validation.
- Insert row/column between the conditional formatting applied cells.
- Insert/delete row/column between the filter applied cells.

Hide and show

You can show or hide the rows and columns in the spreadsheet through property binding, method, and context menu.

Row

The rows can be hidden or shown through the following ways,

- Using `hidden` property in row, you can hide/show the rows at initial load.
- Using `hideRow` method, you can hide the rows by specifying the start and end row index, set the last argument `hide` as `false` to unhide the hidden rows.
- Right-click on the row header and select the desired option from context menu

Column

The columns can be hidden or shown through following ways,

- Using `hidden` property in columns, you can hide/show the columns at initial load.
- Using `hideColumn` method, you can hide the columns by specifying the start and end column index, set the last argument `hide` as `false` to unhide the hidden columns.
- Right-click on the column header and select the desired option from context menu

The following code example shows the hide/show rows and columns operation in the spreadsheet.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
 '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Hiding the 1st and 2nd column index through property binding
let columns: ColumnModel[] = [{ width: 150 }, { width: 100, hidden: true },
 { width: 100, hidden: true }, { width: 80 }, { width: 80 }, { width: 80 }, {
 width: 80 }, { width: 80 }];
// Hiding the 2nd and 3rd row index through property binding
let rows: RowModel[] = [{ index: 2, hidden: true }, { hidden: true }];
let sheets: SheetModel[] = [{ ranges: [{ dataSource: data }], columns:
 columns, rows: rows }];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,

```

```

created: (): void => {
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
    'A1:H1');
    // Unhide the 2nd index hidden column
    spreadsheet.hideColumn(1, 1, false);
    // Unhide the 3rd index hidden row
    spreadsheet.hideRow(3, 3, false);
    // Hiding the 6th index column
    spreadsheet.hideColumn(6);
    // Hiding the 8th and 9th index row
    spreadsheet.hideRow(8, 9);
    spreadsheet.cellFormat({ textAlign: 'center' }, 'D2:H11');
},
// Removed the unwanted support for this samples
showRibbon: false, showFormulaBar: false, showSheetTabs: false
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Size

You can change the size of rows and columns in the spreadsheet by using [setRowsHeight](#) and [setColumnsWidth](#) methods.

Row

You can change the height of single or multiple rows by using the [setRowsHeight](#) method.

You can provide the following type of ranges to the method:

- Single row range: ['2:2']
- Multiple rows range: ['1:100']
- Multiple rows with discontinuous range: ['1:10', '15:25', '30:40']
- Multiple rows with different sheets: [Sheet1!1:50, 'Sheet2!1:50', 'Sheet3!1:50']

The following code example shows how to change the height for single/multiple rows in the spreadsheet.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
'@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [{ ranges: [{ dataSource: data }] }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheets,
    created: (): void => {
        // To change height for single row
        spreadsheet.setRowsHeight(40, ['2']);
        // To change height for multiple rows
        spreadsheet.setRowsHeight(40, ['4:8', '10:12']);
    },
},

```

```
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column

You can change the width of single or multiple columns by using the [setColumnsWidth](#) method.

You can provide the following type of ranges to the method:

- Single column range: ['F:F']
- Multiple columns range: ['A:F']
- Multiple columns with discontinuous range: ['A:C', 'G:I', 'K:M']
- Multiple columns with different sheets: [Sheet1!A:H, 'Sheet2!A:H', 'Sheet3!A:H']

The following code example shows how to change the width for single/multiple columns in the spreadsheet.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel, RowModel } from
 '@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let sheets: SheetModel[] = [{ ranges: [{ dataSource: data }] }];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,
  created: (): void => {
    // To change width of single column
    spreadsheet.setColumnsWidth(100, ['F']);
    // To change width of multiple columns
    spreadsheet.setColumnsWidth(120, ['A:C', 'G:I', 'K:M']);
  },
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing text in column headers

Using the [beforeCellRender](#) event, you can change the text in the column headers. In that event, you can use the `e-header-cell` class to identify the header cell element and update its text value.

The following code example shows how to change the text in the column headers.

INDEX.TS

```

import { Spreadsheet, CellRenderEventArgs } from '@syncfusion/ej2-
spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);

```

```

let spreadsheet: Spreadsheet = new Spreadsheet({
  beforeCellRender: (args: CellRenderEventArgs) => {
    if (
      args.colIndex >= 0 &&
      args.colIndex <= 10 &&
      args.element.classList.contains('e-header-cell')
    ) {
      let text: string = 'custom header ' + args.colIndex.toString();
      args.element.innerText = text;
    }
  }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="JavaScript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

See Also

- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)

Filter in EJ2 JavaScript Spreadsheet control

Filtering helps you to view specific rows in the spreadsheet by hiding the other rows. You can use the [allowFiltering](#) property to enable or disable filtering functionality.

* The default value for `allowFiltering` property is `true`.

By default, the `Filter` module is injected internally into Spreadsheet to perform filtering.

Apply filter on UI

In the active sheet, select a range of cells to filter by value of the cell. The filtering can be done by any of the following ways:

- Select the filter item in the Ribbon toolbar.
- Right-click the sheet, select the filter item in the context menu.
- Use the [applyFilter\(\)](#) method programmatically.
- Use `Ctrl + Shift + L` keyboard shortcut to apply the filter.

* Use `Alt + Up/Down` keyboard shortcut to open the filter dialog.

Filter by criteria

The [applyFilter\(\)](#) method will apply the filter UI, based on the predicate and range given in the arguments.

* The [beforeFilter](#) event will be triggered before filtering the specified range.

* The [filterComplete](#) event will be triggered after the filter action is completed successfully.

The following code example shows `filter` functionality in the Spreadsheet control.

INDEX.TS

```
import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { tradeData } from './datasource.ts';
let columns: ColumnModel[] = [{ width: 100 }, { width: 130 }, { width: 96 },
    { width: 130 }, { width: 130 }, { width: 96 },
    { width: 100 }, { width: 100 }, { width: 110 }, { width: 100 }, { width:
130 }, { width: 150 }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ ranges: [{ dataSource: tradeData }], columns: columns }],
    allowFiltering: true,
    dataBound: function () {
        if (spreadsheet.activeSheetIndex === 0) {
            let departments: string[] = ['Sweden', 'Canada', 'UK'];
            let predicateList: PredicateModel[] = []
            departments.forEach((department: string) => {
                predicateList.push({ field: 'D', predicate: 'or', operator: 'equal', value:
                department });
            });
            spreadsheet.applyFilter(predicateList);
        }
    });
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Filter by cell value

To apply a filter for a cell value, right-click the cell and choose filter -> **Filter By Selected Cell's Value** option from the menu. It applies the filter based on the value of the selected cell in the current sheet.

Clear filter

After applying filter to a certain column, you may want to clear it to make all filtered rows visible again. It can be done in the following ways,

- Choose **Clear** option in ribbon toolbar under **Filter and Sort**. It clears the filters applied in the spreadsheet for all fields.
- Use the [clearFilter\(\)](#) method programmatically, to clear the applied filters in spreadsheet for all fields.

Clear filter on a field

After filtering, you can clear/reset the filter for a field alone. It can be done in the following ways,

- Click filter icon in the column's header and then choose **Clear Filter** option from the filter dialog.
- You can right-click on a filtered column cell and choose **Clear Filter from** . option from the context menu.
- Use the [clearFilter\(field\)](#) method programmatically, to clear the filter in a particular column.

Reapply filter

When you want to reapply the filter after some changes happened in the rows. It can be done in the following ways,

- You can choose **Reapply** option in ribbon toolbar under **Filter and Sort** to reapply the filtered columns again.
- You can right-click on a filtered cell and choose **Reapply** option from the context menu. It reapplies the filters again in the Spreadsheet for all the fields.

Known error validations

The following errors have been handled for filtering,

- *Out of range validation:* When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message **Select a cell or range inside the used range and try again** will be displayed. No filter will be performed if the range is invalid.

Limitations

The following features have some limitations in Filter:

- Insert/delete row/column between the filter applied cells.
- Merge cells with filter.
- Copy/cut paste the filter applied cells.

See Also

- [Sorting](#)
- [Hyperlink](#)
- [Undo Redo](#)

Sort in EJ2 JavaScript Spreadsheet control

Sorting helps arranging the data to a specific order in a selected range of cells. You can use the [allowSorting](#) property to enable or disable sorting functionality.

* The default value for [allowSorting](#) property is **true**.

By default, the **sort** module is injected internally into Spreadsheet to perform sorting.

Sort by cell value

In the active Spreadsheet, select a range of cells to sort by cell value. The range sort can be done by any of the following ways:

- Select the sort item in the Ribbon toolbar and choose the ascending or descending item.
- Right-click the sheet, select the sort item in the context menu and choose the ascending/descending item.
- Use the [sort\(\)](#) method programmatically.

The cell values can be sorted in the following orders:

- Ascending
- Descending

* Ascending is the default order for sorting.

The `sort()` method with empty arguments will sort the selected range by active cell's column as sort column in ascending order.

* The `beforeSort` event will be triggered before sorting the specified range.

* The `sortComplete` event will be triggered after the sort action is completed successfully.

The following code example shows `Sort` functionality in the Spreadsheet control.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from './datasource.ts';
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [{ ranges: [{ dataSource: defaultData }] }],
  allowSorting: true,
  beforeSort: function (args) {
    //code here to handle sorting arguments.
  },
  sortComplete: function (args) {
    spreadsheet.selectRange(args.range);
    // code here.
  },
  dataBound: function () {
    if (spreadsheet.activeSheetIndex === 0) {
      spreadsheet.sort({ containsHeader: true }, 'A1:H11');
    }
  }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Data contains header

You can specify whether the selected range of cells contains header. To specify, you need to set the [containsHeader](#) property to `true` and pass it as `sortOption` arguments of the `sort()` method.

* If the `containsHeader` property is not set and active cell column's first cell value type is differed from the second cell value type, the first row data in the range are marked as column headers.

You can also enable or disable this property using `beforeSort` event arguments,

```
`ts
```

```
let spreadsheet: Spreadsheet = new Spreadsheet({
  beforeSort: function (args) {
    args.sortOptions.containsHeader = true;
```

```

}
});
`

```

In the custom sort dialog, the **Data contains header** checkbox is checked on load. Thus, the default value for **containsHeader** is **true** in custom sort dialog.

Case sensitive sort

The default sort functionality of Spreadsheet is a case insensitive sorting. When you want to perform sorting with case sensitive, you need to set the [caseSensitive](#) property to **true** and pass it as **sortOption** arguments of the `sort()` method.

Case sensitive sorting is applicable only for cells with alphabets. In ascending order sorting with case sensitive enabled, the cells with lower case text will be placed above the cells with upper case text.

* The default value for the **caseSensitive** property is **false**.

You can also enable or disable this property using **beforeSort** event arguments,

```
`ts
```

```

let spreadsheet: Spreadsheet = new Spreadsheet({
  beforeSort: function (args) {
    args.sortOptions.caseSensitive = true;
  }
});
`

```

In the custom sort dialog, the **Case sensitive** checkbox is unchecked on load as the default value is **false**.

Sort multiple columns

When you want to perform sorting on multiple columns, it can be done by any of the following ways:

- Select the **Custom sort...** menu item from the Ribbon toolbar item or context menu item.
- Use the `sort()` method programmatically by providing sort criteria.

* The current sorting functionality supports sorting based on cell values only.

Custom sort dialog

The custom sort dialog helps sorting multiple columns in the selected range by utilizing the rich UI. This dialog will be appeared while choosing the **Custom sort...** from the Ribbon item or context menu item. By default, sort criteria with the first column name from the selected range will be appeared in the dialog on initial load and it cannot be removed.

You can add multiple criteria using the **Add Column** button at the bottom of the dialog. Thus, multiple columns can be specified with different sort order. The newly added sort criteria items can be removed using the **delete** icons at the end of each items.

You can refer to the [Data contains header](#) topic to learn more about **Data contains header** checkbox. To learn more about **Case sensitive** checkbox, you can refer to [Case sensitive sort](#) topic.

Passing sort criteria manually

The multi-column sorting can also be performed manually by passing sort options to the **sort()** method programmatically. The **sortOption** have the following arguments:

- [sortDescriptors](#) – Sort criteria collection that holds the collection of field name, sort order, and [sortComparer](#).
- **containsHeader** – Boolean argument that specifies whether the range has headers in it.
- **caseSensitive** – Boolean argument that specifies whether the range needs to consider case.

* All the arguments are optional.

* When a **sortDescriptor** is specified without field, the field of the first **sortDescriptor** from the collection will be assigned from active cell's column name and others will be ignored. Hence, it will act as single column sorting.

INDEX.TS

```
import { Spreadsheet, SortDescriptor } from '@syncfusion/ej2-spreadsheet';
import { DataManager } from '@syncfusion/ej2-data';
import { tradeData } from './datasource.ts';
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [{ ranges: [{ dataSource: tradeData }] }],
  allowSorting: true,
  sortComplete: function (args) {
    spreadsheet.selectRange(args.range);
    // code here.
  },
  dataBound: function () {
    let sortDescriptors: SortDescriptor[] = [
      {
        field: 'F',
        order: 'Ascending'
      },
      {
        field: 'E',
        order: 'Ascending'
      },
      {
        field: 'C',
        order: 'Descending'
      }
    ];
    if (spreadsheet.activeSheetIndex === 0) {
      spreadsheet.sort({ sortDescriptors: sortDescriptors,
        containsHeader: true }, 'A1:H30');
    }
  }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Custom sort comparer

The [sortDescriptor](#) holds the [sortComparer](#) property, which is a function and it is used to customize the sort comparer for specific sort criteria. Each [sortDescriptor](#) can be customized using the custom sort comparer function.

By customizing sort comparer, you can define the sort action as desired.

* The [sortComparer](#) is an optional property of [sortDescriptor](#).

For custom sort comparer example, refer to the [Sort a range by custom list](#) in the [how-to](#) section.

Known error validations

The following errors have been handled for sorting,

- *Out of range validation:* When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message [Select a cell or range inside the used range and try again](#) will be displayed. No sort will be performed if the range is invalid.
- *Empty field validation:* When the sort criteria does not have a column selected (empty) in the custom sort dialog, it will become invalid, and an error message [Sort criteria column should not be empty](#) will be displayed on [OK](#) button click.
- *Duplicate field validation:* When the column names of added sort criteria are repeated more than once in the custom sort dialog, it will become invalid and an error message [is mentioned more than once. Duplicate columns must be removed](#) will be displayed on [OK](#) button click.

Limitations

- Sorting is not supported with formula contained cells.

See Also

- [Sort a range by custom list](#)
- [Hyperlink](#)
- [Filtering](#)
- [Undo Redo](#)

Link in EJ2 JavaScript Spreadsheet control

Hyperlink is used to navigate to web links or cell reference within the sheet or to other sheets in Spreadsheet. You can use the [allowHyperlink](#) property to enable or disable hyperlink functionality.

* The default value for [allowHyperlink](#) property is [true](#).

Insert Link

You can insert a hyperlink in a worksheet cell for quick access to related information.

User Interface:

In the active spreadsheet, click the cell where you want to create a hyperlink. Insert hyperlink can be done by any of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the **Link** item.
- Right-click the cell and then click Hyperlink item in the context menu.
- Use **Ctrl + K** keyboard shortcut to apply the hyperlink.
- Use the [addHyperlink](#) method programmatically.

Edit Hyperlink

You can change an existing hyperlink in your workbook by changing its destination or the text that is used to represent it.

User Interface:

In the active spreadsheet, Select the cell that contains the hyperlink that you want to change. Editing the hyperlink while opening the dialog can be done by any one of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the **Link** item.
- Right-click the cell and then click Edit Hyperlink item in the context menu, or you can press **Ctrl+K**.

In the Edit Link dialog box, make the changes that you want, and click UPDATE.

Remove Hyperlink

Performing this operation remove a single hyperlink without losing the display text.

User Interface:

In the active spreadsheet, click the cell where you want to remove a hyperlink. remove hyperlink can be done by any of the following ways:

- Right-click the cell and then click Remove Hyperlink item in the context menu.
- Use the [removeHyperlink\(\)](#) method programmatically.

How to change target attribute

There is an event named **beforeHyperlinkClick** which triggers only on clicking hyperlink. You can customize where to open the hyperlink by using the **target** property in the arguments of that event.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
let sheet: SheetModel[] = [
  {
    name: 'PriceDetails'
    rows: [
      {
        cells: [
          { value: 'Item Name' },
          { value: 'Stock Detail' },
          { value: 'Website' }
        ]
      },
      {
        cells: [
          { value: 'Casual Shoes' },
          { value: 'OUT OF STOCK' },
        ]
      }
    ]
  }
];
```

```

        { value: 'Amazon', hyperlink: 'https://www.amazon.com/'
    }
    ],
    },
    {
        cells: [
            { value: 'Sports Shoes' },
            { value: 'IN STOCK', hyperlink: 'Stock!A2:B2' },
            { value: 'Overstack', hyperlink:
'https://www.overstock.com/' }
        ]
    },
    {
        cells: [
            { value: 'Formal Shoes' },
            { value: 'IN STOCK', hyperlink: 'Stock!A3:B3' },
            { value: 'AliExpress', hyperlink:
'https://www.aliexpress.com/' }
        ]
    },
    {
        cells: [
            { value: 'Sandals & Floaters' },
            { value: 'OUT OF STOCK' },
            { value: 'AliBaba', hyperlink: 'http://www.alibaba.com/'
}
        ]
    },
    {
        cells: [
            { value: 'Flip-Flops & Slippers' },
            { value: 'IN STOCK', hyperlink: 'Stock!A4:B4' },
            { value: 'Taobao', hyperlink: 'https://taobao.com/' }
        ]
    },
    ],
    columns: [
        { width: 160 }, { width: 160 }, { width: 120 }
    ]
},
{
    name: 'Stock',
    rows: [
        {
            cells: [
                { value: 'Item Name' },
                { value: 'Available Count' },
            ]
        },
        {
            cells: [
                { value: 'Sports Shoes' },
                { value: '30' },
            ]
        },
        {
            cells: [

```

```

        { value: 'Formal Shoes' },
        { value: '25' },
    ],
},
{
    cells: [
        { value: 'Flip-Flops & Slippers' },
        { value: '40' },
    ],
},
{
    cells: [
        { value: 'Running Shoes' },
        { value: '25' },
    ],
},
],
columns: [
    { width: 160 }, { width: 120 }
]
}
];
//Initialize the Spreadsheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: sheet,
    beforeHyperlinkClick: (args: BeforeHyperlinkArgs) => {
        args.target = '_self'; //change target attribute
    }
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations

- Inserting hyperlink not supported for multiple ranges.

See Also

- [Sorting](#)
- [Filtering](#)
- [Undo Redo](#)

Clipboard in EJ2 JavaScript Spreadsheet control

The Spreadsheet provides support for the clipboard operations (cut, copy, and paste). Clipboard operations can be enabled or disabled by setting the [enableClipboard](#) property in Spreadsheet.

By default, the `enableClipboard` property is true.

Cut

It is used to cut the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

User Interface:

Cut can be done in one of the following ways.

- Using Cut button in the Ribbon's HOME tab to perform cut operation.
- Using Cut option in the Context Menu.
- Using **Ctrl + X** | **Command + X** keyboard shortcut.
- Using the [cut](#) method.

Copy

It is used to copy the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

User Interface:

Copy can be done in one of the following ways.

- Using Copy button in the Ribbon's HOME tab to perform copy operation.
- Using Copy option in the Context Menu.
- Using **Ctrl + C** | **Command + C** keyboard shortcut.
- Using the [copy](#) method.

Paste

It is used to paste the clipboard data to the selected range, rows or columns. You have the following options in Paste,

- **Paste Special** - You can paste the values with formatting.
- **Paste** - You can paste only the values without formatting.

It also performs for external clipboard operation. If you perform cut and paste, clipboard data will be cleared, whereas in copy and paste the clipboard contents will be maintained. If you perform paste inside the copied range, the clipboard data will be cleared.

User Interface:

Paste can be done in one of the following ways.

- Using Paste button in the Ribbon's HOME tab to perform paste operation.
- Using Paste option in the Context Menu.
- Using **Ctrl + V** | **Command + V** keyboard shortcut.
- Using the [paste](#) method.

If you use the Keyboard shortcut key for cut (**Ctrl + X**) | copy (**Ctrl + C**) from other sources, you should use **Ctrl + V** shortcut while pasting into the spreadsheet.

INDEX.TS

```
import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from './datasource.ts';
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
//Initialize action items.
let items: ItemModel[] = [
  {
```

```

        text: "Copy"
    },
    {
        text: "Cut"
    },
    {
        text: "Paste"
    }
];
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton({
    items: items,
    cssClass: "e-round-corner",
    select: (args: MenuEventArgs) => {
        if (args.item.text === 'Copy')
            spreadsheet.copy();
        if (args.item.text === 'Cut')
            spreadsheet.cut();
        if (args.item.text === 'Paste')
            spreadsheet.paste();
    }
});
// Render initialized DropDownButton.
drpDownBtn.appendTo("#element");
let columns: ColumnModel[] = [
    {
        width: 130
    },
    {
        width: 92
    },
    {
        width: 96
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Price Details', ranges: [{ dataSource: defaultData }] },
    columns: columns ],
    enableClipboard: true,
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:F1');
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

        <link rel="shortcut icon" href="resources/favicon.ico">
        <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
        <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
        <link href="styles.css" rel="stylesheet">

        <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
        <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <button id="element">Clipboard</button>
    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent the paste functionality

The following example shows, how to prevent the paste action in spreadsheet. In [actionBegin](#) event, you can set `cancel` argument as false in paste request type.

INDEX.TS

```

import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from './datasource.ts';
import { DropDownButton, ItemModel } from "@syncfusion/ej2-splitbuttons";
//Initialize action items.
let items: ItemModel[] = [
    {
        text: "Copy"
    },
    {
        text: "Cut"
    },
    {
        text: "Paste"
    }
];
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton({
    items: items,
    cssClass: "e-round-corner",
    select: (args: MenuEventArgs) => {
        if (args.item.text === 'Copy')
            spreadsheet.copy();
        if (args.item.text === 'Cut')
            spreadsheet.cut();
        if (args.item.text === 'Paste')
            spreadsheet.paste();
    }
});
// Render initialized DropDownButton.
drpDownBtn.appendTo("#element");
let columns: ColumnModel[] = [
    {
        width: 130
    },
    {
        width: 92
    },
    {
        width: 96
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Price Details', ranges: [{ dataSource: defaultData }] },
    columns: columns }],
    enableClipboard: true,
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:F1');
    },
    // Triggers before the action begins.
    actionBegin: (pasteArgs: any) => {
        const args: { action: string, eventArgs: BeforePasteEventArgs } = {
            action: pasteArgs.args.eventArgs.requestType,
            eventArgs: pasteArgs.args.eventArgs };
        // To cancel the paste action.

```

```

        if (args.action === 'paste') {
            args.eventArgs.cancel = true;
        }
    });
    //Render the initialized Spreadsheet
    spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <button id="element">Clipboard</button>

```

```
<div id="container">
  <div id="spreadsheet"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Limitations

- External clipboard is not fully supported while copying data from another source and pasting into a spreadsheet, it only works with basic supports (Values, Number, cell, and Text formatting).
- If you copy =SUM(A2,B2) and paste, the formula reference will change depending on the pasted cell address but we don't have support for nested formula(formula reference will be same).
- Clipboard is not supported with conditional formatting (values only pasting).
- We have limitation while copying the whole sheet data and pasting it into another sheet.

Scrolling in EJ2 JavaScript Spreadsheet control

Scrolling helps you to move quickly to different areas of the worksheet. It moves faster if we use horizontal and vertical scroll bars. Scrolling can be enabled by setting the [allowScrolling](#) as true.

By default, the `allowScrolling` property is true.

You have the following options in Scrolling by using [scrollSettings](#).

- Finite scrolling.
- Virtual scrolling.

Finite Scrolling

Finite scrolling supports two type of modes in scrolling. You can use the [isFinite](#) property in [scrollSettings](#) to specify the mode of scrolling.

- Finite - This mode does not create a new row/column when the scrollbar reaches the end. This can be achieved by setting the [isFinite](#) property as `true`.
- Infinite - This mode creates a new row/column when the scrollbar reaches the end. This can be achieved by setting the [isFinite](#) property as `false`.

By Default, the `isFinite` property is `false`.

Virtual Scrolling

- Virtual scrolling allows you to load data that you require (load data based on viewport size) without buffering the entire huge database. You can set the `enableVirtualization` property in `scrollSettings` as `true`.

In virtual scrolling `enableVirtualization` is set to true means, it allows you to load the spreadsheet data while scrolling.

By Default, the `enableVirtualization` property is `true`.

User Interface:

You can scroll through the worksheet using one of the following ways,

- Using the `arrow` keys.
- Using the Horizontal and Vertical `scroll` bars.
- Using the `mouse` wheel.

Finite scrolling with defined rows and columns

If you want to perform scrolling with defined rows and columns, you must define `rowCount` and `colCount` in the `sheets` property and set `isFinite` as true and `enableVirtualization` as false in `scrollSettings`.

The following code example shows the finite scrolling with defined rows and columns in the spreadsheet. Here, we used `rowCount` as 20 and `colCount` as 20, after reaching the 20th row or 20th column you can't able to scroll.

INDEX.TS

```
import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from '../datasource.ts';
let columns: ColumnModel[] = [
    {
        width: 100
    },
    {
        width: 92
    },
    {
        width: 96
    },
    {
        width: 110
    },
    {
        width: 92
    },
    {
        width: 96
    },
    {
        width: 96
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Price Details', ranges: [{ dataSource: defaultData }] },
    columns: columns, rowCount: 9, colCount: 7 ]},
    allowScrolling: true,
    scrollSettings: { isFinite: true },
    created: (): void => {
```

```

        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:G1');
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>

```

```
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Selection in EJ2 JavaScript Spreadsheet control

Selection provides interactive support to highlight the cell, row, or column that you select. Selection can be done through Mouse, Touch, or Keyboard interaction. To enable selection, set **mode** as **Single** | **Multiple** in [selectionSettings](#). If you set **mode** to **None**, it disables the UI selection.

* The default value for **mode** in **selectionSettings** is **Multiple**.

You have the following options in Selection,

- Cell selection
- Row selection
- Column selection

Cell selection

Cell selection is used to select a single or multiple cells. It can be performed using the [selectRange](#) method.

User Interface:

- Click on a cell to select it (or) use the **arrow** keys to navigate to it and select it.
- To select a range, select a cell, then use the left mouse button to select and drag over to other cells (or) use the **Shift + arrow** keys to select the range.
- To select non-adjacent cells and cell ranges, hold **Ctrl** and select the cells.

You can quickly locate and select specific cells or ranges by entering their names or cell references in the Name box, which is located to the left of the formula bar, and also select named or unnamed cells or ranges by using the Go To (**Ctrl+G**) command.

Row selection

Row selection is used to select a single or multiple rows.

User Interface:

You can perform row selection in any of the following ways,

- By clicking the row header.
- To select multiple rows, select a row header with the left mouse button and drag over to other row headers (or) use the **Shift + arrow** keys to select multiple rows.
- To select non-adjacent rows, hold **Ctrl** and select the row header.
- You can also use the [selectRange](#) method for row selection.

The following sample shows the row selection in the spreadsheet, here selecting the 5th row using the `selectRange` method.

INDEX.TS

```
import { Spreadsheet, ColumnModel, getRangeAddress } from '@syncfusion/ej2-spreadsheet';
import { budgetData } from './datasource.ts';
let columns: ColumnModel[] = [
    {
        width: 130
    },
    {
        width: 92
    },
    {
        width: 96
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData }] },
    columns: columns }],
    selectionSettings: { mode: 'Multiple' },
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:D1');
        let colCount: number = spreadsheet.getActiveSheet().colCount;
        spreadsheet.selectRange(getRangeAddress([4, 0, 4, colCount]));
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column selection

Column selection is used to select a single or multiple columns.

User Interface:

You can perform column selection in any of the following ways,

- By clicking the column header.
- To select multiple columns, select a column header with the left mouse button and drag over to other column headers (or) use the **Shift + arrow** keys to select the multiple columns.
- To select non-adjacent columns, hold **Ctrl** and select the column header.
- You can also use the **selectRange** method for row selection.

The following sample shows the column selection in the spreadsheet, here selecting the 3rd column using the **selectRange** method.

INDEX.TS


```
import { Spreadsheet, ColumnModel, getRangeAddress } from '@syncfusion/ej2-spreadsheet';
import { budgetData } from './datasource.ts';
let columns: ColumnModel[] = [
    {
        width: 130
    },
    {
        width: 92
    },
    {
        width: 96
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData }] },
    columns: columns ],
    selectionSettings: { mode: 'Multiple' },
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle' }, 'A1:D1');
        let rowCount: number = spreadsheet.getActiveSheet().rowCount;
        spreadsheet.selectRange(getRangeAddress([0, 2, rowCount, 2]));
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

How to remove selection in the spreadsheet

The following sample shows, how to remove the selection in the spreadsheet. Here changing the **mode** as **None** in [selectionSettings](#) to disable's the UI selection.

INDEX.TS

```

import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { budgetData } from './datasource.ts';
let columns: ColumnModel[] = [
    {
        width: 130
    },
    {
        width: 92
    },
    {
        width: 96
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData }] },
    columns: columns },
    selectionSettings: { mode: 'None' },
    created: (): void => {

```

```

        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:D1');
    },
    // Triggers before going to the editing mode.
    cellEdit: (args: CellEditEventArgs): void => {
        args.cancel = true;
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

        <!--Element which is going to render-->

        <div id="container">
        <div id="spreadsheet"></div>
        </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

- We have a limitation while performing the Select All(**ctrl + A**). You can do this only by clicking the Select All button at the top left corner.

Protect sheet in EJ2 JavaScript Spreadsheet control

Sheet protection helps you to prevent the users from modifying the data in the spreadsheet.

Protect Sheet

Protect sheet feature helps you to prevent the unknown users from accidentally changing, editing, moving, or deleting data in a spreadsheet. And you can also protect the sheet with password.

You can use the [isProtected](#) property to enable or disable the Protecting functionality.

The default value for **isProtected** property is **false**.

By default in protected sheet, selecting, formatting, inserting, deleting functionalities are disabled. To enable some of the above said functionalities

the **protectSettings** options are used in a protected spreadsheet.

The available **protectSettings** options in spreadsheet are,

Options	Uses
-----	-----
Select Cells	Used to perform Cell Selection.
Format Cells	Used to perform Cell formatting.
Format Rows	Used to perform Row formatting.
Format Columns	Used to perform Column formatting.
Insert Link	Used to perform Hyperlink Insertions.

The default value for all **protectSettings** options are **false**.

By default, the **Protect Sheet** module is injected internally into the Spreadsheet to perform sheet protection function.

User Interface:

In the active Spreadsheet, the sheet protection can be done by any of the following ways:

- Select the Protect Sheet item in the Ribbon toolbar under the Data Tab, and then select your desired options.
- Right-click the sheet tab, select the Protect Sheet item in the context menu, and then select your desired options.
- Use the [protectSheet\(\)](#) method programmatically.

The following example shows **Protect Sheet** functionality with password in the Spreadsheet control.

INDEX.TS

```
import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { budgetData, salaryData } from './datasource.ts';
let columns: ColumnModel[] = [{ width: 100 }, { width: 100 }, { width: 100 },
    { width: 100 }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData }] },
    columns: columns, isProtected: true, protectSettings: { selectCells: true } },
    { name: 'Salary', ranges: [{ dataSource: salaryData }] },
    columns: columns}],
    dataBound: function () {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold' }, 'A11:D11');
        spreadsheet.protectSheet(1, { selectCells: false },
"syncfusion"); // protect sheet with password
    }
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Spreadsheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="dialog"></div>

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations of Protect sheet

- Password encryption is not supported

Unprotect Sheet

Unprotect sheet is used to enable all the functionalities that are already disabled in a protected spreadsheet.

User Interface:

In the active Spreadsheet, the sheet Unprotection can be done by any of the following ways:

- Select the **Unprotect Sheet** item in the Ribbon toolbar under the Data Tab.
- Right-click the sheet tab, select the **Unprotect Sheet** item in the context menu.
- Use the [unprotectSheet\(\)](#) method programmatically.

Unlock the particular cells in the protected sheet

In protected spreadsheet, to make some particular cell or range of cells are editable, you can use [lockCells\(\)](#) method, with the parameter `range` and `isLocked` property as false.

INDEX.TS

```
import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { Button } from '@syncfusion/ej2-buttons';
import { Dialog } from '@syncfusion/ej2-popups';
import { budgetData, salaryData } from './datasource.ts';
let columns: ColumnModel[] = [{ width: 100 }, { width: 100 }, { width: 100 },
    { width: 100 }]
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData }] },
    columns: columns, isProtected: true, protectSettings: { selectCells: true } },
    { name: 'Salary', ranges: [{ dataSource: salaryData }] },
    columns: columns}],
    dataBound: function () {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold' }, 'A11:D11');
    }
});
spreadsheet.appendTo('#spreadsheet');
let dialogObj: Dialog = new Dialog({
    header: 'Spreadsheet',
    target: document.getElementById('spreadsheet'),
    content: '"A1:F3" range of cells has been unlocked.',
    showCloseIcon: true,
    isModel: true,
    visible: false,
    width: '500px',
    buttons: [{
        click: lockCells,
        buttonModel: { content: 'Ok', isPrimary: true }
    }
    ]});
dialogObj.appendTo('#dialog');
let button: Button = new Button({ content: 'Unlock cells' });
button.appendTo('#button');
document.getElementById('button').onclick = (): void => {
    dialogObj.show();
}
function lockCells(): void {
    spreadsheet.lockCells('A1:F3', false);
    dialogObj.hide();
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->
    <div id="dialog"></div>

    <div id="container">
        <div id="button"></div>
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```


Protect Workbook

Protect workbook feature helps you to protect the workbook so that users cannot insert, delete, rename, hide the sheets in the spreadsheet.

You can use the [password](#) property to protect workbook with password.

You can use the [isProtected](#) property to protect or unprotect the workbook without the password.

The default value for `isProtected` property is `false`.

User Interface:

In the active Spreadsheet, you can protect the worksheet by selecting the Data tab in the Ribbon toolbar and choosing the **Protect Workbook** item. Then, enter the password and confirm it and click on OK.

The following example shows **Protect Workbook** by using the [isProtected](#) property in the Spreadsheet control.

INDEX.TS

```
import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { budgetData } from './datasource.ts';
let columns: ColumnModel[] = [{ width: 100 }, { width: 100 }, { width: 100 },
    { width: 100 }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    // To protect the workbook
    isProtected: true,
    sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData } ]},
    columns: columns }],
    dataBound: function () {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
    }
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following example shows **Protect Workbook** by using the [password](#) property in the Spreadsheet control. To unprotect the workbook, click the unprotect workbook button in the data tab and provide the password as syncfusion in the dialog box.

INDEX.TS

```

import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { budgetData } from './datasource.ts';
let columns: ColumnModel[] = [{ width: 100 }, { width: 100 }, { width: 100 }, { width: 100 }];
let spreadsheet: Spreadsheet = new Spreadsheet({
  //To protect the workbook with password
  password: "syncfusion",
  sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData } ]},
  columns: columns }],
  dataBound: function () {

```

```

        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
    }
}
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">

```

```
<div id="spreadsheet"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Unprotect Workbook

Unprotect Workbook is used to enable the insert, delete, rename, move, copy, hide or unhide sheets feature in the spreadsheet.

User Interface:

In the active Spreadsheet, the workbook Unprotection can be done in any of the following ways:

- Select the **Unprotect Workbook** item in the Ribbon toolbar under the Data Tab and provide the valid password in the dialog box.

See Also

- [Hyperlink](#)

Searching in EJ2 JavaScript Spreadsheet control

Find and Replace helps you to search for the target text and replace the found text with alternative text within the sheet or workbook. You can use the [allowFindAndReplace](#) property to enable or disable the Find and Replace functionality.

* The default value for **allowFindAndReplace** property is **true**.

Find

Find feature is used to select the matched contents of a cell within the sheet or workbook. It is extremely useful when working with large set of data source.

User Interface:

Find can be done by any of the following ways:

- Select the Search icon in the Ribbon toolbar or use **Ctrl + F** key to open the Find dialog.
- Use find Next and find Previous buttons to search the given value in the workbook.
- Select the option button in Find dialog to open the Find and Replace dialog. Then, select the below properties for enhanced searching.

* **Search within**: To search the target in a sheet (default) or in an entire workbook.

* **Search by**: It enhance the search, either By Rows (default), or By Columns.

* **Match case**: To find the matched value with case sensitive.

* **Match exact cell contents:** To find the exact matched cell value with entire match cases.

- Using [find\(\)](#) method to perform find operation.

Replace

Replace feature is used to change the find contents of a cell within sheet or workbook. Replace All is used to change all the matched contents of a cell within sheet or workbook.

User Interface:

Replace can be done by any of the following ways:

- Use **Ctrl + H** key to open the Find and Replace dialog.
- Use Replace button to change the found value in sheet or workbook.
- Using Replace All button, all the matched criteria can be replaced with find value based on sheet or workbook.
- Using [replace\(\)](#) method to perform replace operation by passing the argument `args.replaceby` as `replace`.
- Using [replace\(\)](#) method to perform replace all operation by passing the argument `args.replaceby` as `replaceall`.

Go to

Go to feature is used to navigate to a specific cell address in the sheet or workbook.

User Interface:

- Use **Ctrl + G** key to open the Go To dialog.
- Use [goTo\(\)](#) method to perform Go To operation.

In the following sample, searching can be done by following ways:

- Select the Home tab in the Ribbon toolbar, and then select the Search icon.
- Enter any value in the search textbox.
- Select the next (or) previous button to find the entered value in the spreadsheet.
- You can have more options to find values by selecting the more options in the search toolbar.

INDEX.TS

```
import { Spreadsheet, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData, budgetData } from '../datasource.ts';
let columns: ColumnModel[] = [
    {
        width: 130
    },
    {
        width: 92
    },
    {
        width: 96
    }
];
```

```
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [{ name: 'Price Details', ranges: [{ dataSource: defaultData }],
  columns: columns },
  { name: 'Budget', ranges: [{ dataSource: budgetData }], columns: columns
}],
  allowFindAndReplace: true
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
```

```

        <div id="container">
        <div id="spreadsheet"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Limitations

- Undo/redo for Replace All is not supported in this feature.

Keyboard shortcuts in EJ2 JavaScript Spreadsheet control

The keyboard shortcuts supported in the spreadsheet are,

Shortcut	Description
----- -----	
Ctrl + O	Displays dialog to open a file.
Ctrl + S / Alt + F2	Saves the workbook.
F2	Enables edit mode.
ESC	Cancel edit mode and discard the changes.
Backspace and SPACE	Clears content of the active cell and enables edit mode.
Ctrl + C	Copies the selected cells.
Ctrl + X	Cuts the selected cells.
Ctrl + V	Paste the clipboard(cut or copied) content in the new selected range.
Ctrl + B	Applies or removes bold formatting.
Ctrl + I	Applies or removes <i>italic</i> formatting.
Ctrl + U	Applies or removes <u>underline</u> .
Ctrl + 5	Applies or removes strikethrough .
Ctrl + Z	Reverses (Undo) the last action.
Ctrl + Y	Repeats (Redo) the last reversed action.
Ctrl + K	It opens the Insert Hyperlink dialog for adding new hyperlink to a cell. If the selected cell already contains hyperlink, it opens the Edit Hyperlink dialog.
Ctrl + F / Shift + F5	Opens Find dialog.
Ctrl + H	Opens Find and Replace dialog.

- | Ctrl + G | Opens **GoTo** dialog, which helps to navigate to cell. |
- | Ctrl + Shift + L | Applies filter to the first row of the selected range or used range. |
- | Alt + F | Opens the **File** menu. |
- | Alt + H | Go to **Home** tab. |
- | Alt + N | Go to **Insert** tab. |
- | Alt + M | Go to **Formulas** tab. |
- | Alt + A | Go to **Data** tab. |
- | Alt + W | Go to **View** tab. |
- | Tab | Navigate the active cell to the next cell in the same row. |
- | Shift + Tab | Navigate the active cell to the previous cell in the same row. |
- | Right or Left arrow | Move the focus to next or previous item in the tab if the focus is on ribbon tab. |
- | Up arrow | When a menu is open, move focus to the next item. |
- | Down arrow | When a menu is open, move focus to the previous item. |
- | Spacebar or Enter | Activate a selected button. |
- | Ctrl + F8 | Expand or collapse the ribbon content. |
- | Ctrl + Shift + U | Expand or collapse the formula bar. |
- | Ctrl + 9 | Hide the selected row. |
- | Ctrl + 0 | Hide the selected column. |
- | Home | Moves the selection to starting column in worksheet. |
- | Ctrl + Home | Move the selection to the first visible cell on a worksheet. |
- | Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |
- | Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |
- | Ctrl + & | Apply an outline border to the selected cells. |
- | Ctrl + Shift + & | Apply an outline border to the cells that you've chosen. |
- | Ctrl + Shift + ~ | Apply the **General** number format. |
- | Ctrl + Shift + \$ | Apply the **Currency** format with two decimal places (negative numbers in parentheses). |
- | Shift + F10 | Open the context menu. |
- | Ctrl + % | Apply the **Percentage** format with no decimal places. |
- | Ctrl + ^ | Apply the **Scientific** number format with two decimal places. |
- | Ctrl + Shift + # | Apply the **Date** format with the day, month, and year. |
- | Ctrl + Shift + @ | Apply the **Time** format with the hour and minute, and AM or PM. |

| Ctrl + Shift + ! | Apply the **Number** format with two decimal places, thousands separator, and minus sign (-) for negative values. |

| Ctrl + Spacebar | Select an entire column in a worksheet. |

| Shift + Spacebar | Select an entire row in a worksheet. |

| Shift + F3 | Opens **Insert Function** dialog. |

| Ctrl + Alt + N | Opens new workbook. |

| Shift + Page Down | Perform page down by selecting all cells between. |

| Shift + Page Up | Perform page up by selecting all cells between. |

| Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |

| Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |

| Ctrl + Up | Navigate to the first non-blank cell in the same row as the active cell. |

| Ctrl + Down | Navigate to the last cell that is not blank in the same column as the active cell. |

| Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Up | Extend the selection of cells to the first non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Down | Extend the selection of cells to the last non-blank cell in the same row as the active cell. |

| Shift + Alt + K | Open the **List All Sheets** dropdown option. |

| Up arrow | Navigate from the active cell to the previous cell in the same column. |

| Down arrow | Navigate from the active cell to the next cell in the same column. |

| Left arrow | Navigate from the active cell to the previous cell in the same row. |

| Right arrow | Navigate from the active cell to the next cell in the same row. |

| Page Up | Move page up. |

| Page Down | Move page down. |

| Shift + Up | Extend the selection of cells to the previous row. |

| Shift + Down | Extend the selection of cells to the next row. |

| Shift + Left | Extend the selection of cells to the previous column. |

| Shift + Right | Extend the selection of cells to the next column. |

| Ctrl + Top | Navigate to the non-blank cell before the active cell in the same column. |

| Ctrl + Shift + Top | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |

| Ctrl + Shift + Bottom | Extend the cell selection to the last non-blank cell in the same column as the active cell. |

| Enter | Navigate the active cell to the next cell in the same column. |

| Shift + Enter | Navigate to the previous cell in the same column from the active cell. |

| Alt + Down | Open the list data validation menu and filter menu. |

| Alt + Up | Close the list data validation menu and filter menu. |

| Delete | Remove the contents of the cell. |

| Shift + Home | Extend the cell selection to the first column of a worksheet. |

| Shift + F11 | Add a new sheet. |

| Ctrl + Shift + 9 | Unhide the selected rows. |

| Ctrl + Shift + 0 | Unhide the selected columns. |

| Ctrl + D | Fill the cell down. |

| Ctrl + R | Fill the cell right. |

| Alt + Enter | While editing, add a new line. |

| Enter | Complete the cell editing and select the cell below in the same column. |

| Shift + Enter | Complete the cell editing and select the cell above in the same column. |

| Tab | Complete the cell editing and select the next cell in the same row. |

| Shift + Tab | Complete the cell editing and select the previous cell in the same row. |

| Alt | Focus on the active ribbon tab. |

| Left | Move the focus to the previous items in the ribbon content. |

| Right | Move the focus to the next items in the ribbon content. |

| Alt + Down | Open the ribbon dropdown menu. |

| Esc / Alt + Up | Close the ribbon dropdown menu. |

See Also

- [Formatting](#)

Ribbon in EJ2 JavaScript Spreadsheet control

It helps to organize a spreadsheet's features into a series of tabs. By clicking the expand or collapse icon, you can expand or collapse the ribbon toolbar dynamically.

Ribbon Customization

You can customize the ribbon using the following methods,

| Method | Action |

|-----|-----|

| [hideRibbonTabs](#) | Used to show or hide the existing ribbon tabs. |

- | [enableRibbonTabs](#) | Used to enable or disable the existing ribbon tabs. |
- | [addRibbonTabs](#) | Used to add custom ribbon tabs. |
- | [hideToolbarItems](#) | Used to show or hide the existing ribbon toolbar items. |
- | [enableToolbarItems](#) | Used to enable or disable the specified toolbar items. |
- | [addToolbarItems](#) | Used to add the custom items in ribbon toolbar. |
- | [hideFileMenuItems](#) | Used to show or hide the ribbon file menu items. |
- | [enableFileMenuItems](#) | Used to enable or disable file menu items. |
- | [addFileMenuItems](#) | Used to add custom file menu items. |

The following code example shows the usage of ribbon customization.

INDEX.TS

```
import { Spreadsheet, SheetModel, ColumnModel, MenuSelectEventArgs } from
 '@syncfusion/ej2-spreadsheet';
import { enableRipple, select } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { data } from './datasource.ts';
enableRipple(true);
let columns: ColumnModel[] = [{ width: 180 }, { width: 130 }, { width: 130
 }, { width: 180 }, { width: 130 }, { width: 120 }];
let sheets: SheetModel[] = [{ ranges: [{ dataSource: data }], columns:
 columns }];
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: sheets,
  created: (): void => {
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:F1');
    // Hiding the `Insert` from ribbon.
    spreadsheet.hideRibbonTabs(['Insert']);
    // Set disabled state to `View` ribbon tab.
    spreadsheet.enableRibbonTabs(['View'], false);
    // Adding the `Help` ribbon tab at the last index.
    // Specify the ribbon tab header text in last optional
    argument(`insertBefore`) for inserting new tab before any existing tab.
    spreadsheet.addRibbonTabs([{ header: { text: 'Help' }, content: [{
    text: 'Feedback', tooltipText: 'Feedback',
      click: (args: ClickEventArgs): void => { /* Any click action for
    this toolbar item will come here. */ } } ]});
    // Hiding the unwanted toolbar items from `Home` by specifying its
    index.
    spreadsheet.hideToolbarItems('Home', [0, 1, 2, 4, 14, 15, 21, 24]);
    // Set disable state to `Underline`, `Wrap text` toolbar items from
    `Home` by specifying the item id.
    spreadsheet.enableToolbarItems('Home',
    [`${spreadsheet.element.id}_underline`, `${spreadsheet.element.id}_wrap`],
    false);
    // Set disable state to `Protect Sheet` toolbar item from `Data` by
    mentioning its index.
    spreadsheet.enableToolbarItems('Data', [0], false);
    // Adding the new `Custom Formulas` toolbar item under `Formulas`
    tab for adding custom formulas.
```

```

spreadsheet.addToolBarItems(
    'Formulas', [{ type: 'Separator' }, {
        text: 'Custom Formulas', tooltipText: 'Custom Formulas',
        // You can set click handler for each new custom toolbar
item
        click: (args: ClickEventArgs): void => {
            // You can add custom formulas here.
        }
    }]);
    // Adding new custom item `Import` after the existing `Open` item.
    By default, new item will add after the specified item.
    spreadsheet.addFileMenuItems([{ text: 'Import', iconCss: 'e-open e-
icons' }], 'Open');
    // Adding new custom item `Export As` after the existing `Save As`
item.
    // Set `insertAfter` optional argument as `false` for adding new
item before the specified item.
    spreadsheet.addFileMenuItems(
        [{
            text: 'Export As', iconCss: 'e-save e-icons', items: [{
text: 'XLSX', iconCss: 'e-xlsx e-icons' },
                { text: 'XLS', iconCss: 'e-xls e-icons' }, { text:
'CSV', iconCss: 'e-csv e-icons' }]
        }],
        'Save As', false);
    },
    fileMenuBeforeOpen: (): void => {
        // Because the file menu items are created dynamically, you need to
perform the hide or show and enable/disable operations
        // under filemenu before open event.
        // Hiding the `Save As` and `Open` item.
        spreadsheet.hideFileMenuItems(['Save As', 'Open']);
        // Set disable state to `New` item. You can perform any file menu
items customization by specifying the item id,
        // if it has more than one same item text. Set the last argument
`isUniqueId` as true for using the item id.
        spreadsheet.enableFileMenuItems(['${spreadsheet.element.id}_New`,
false, true);
    },
    fileMenuItemSelect: (args: MenuSelectEventArgs): void => {
        // Custom file menu items select handler
        switch (args.item.text) {
            case 'Import': select(`${spreadsheet.element.id}_fileUpload`,
spreadsheet.element).click();
                break;
            case 'XLSX': spreadsheet.save({ saveType: 'Xlsx' });
                break;
            case 'XLS': spreadsheet.save({ saveType: 'Xls' });
                break;
            case 'CSV': spreadsheet.save({ saveType: 'Csv' });
                break;
        }
    },
    openUrl:
'https://services.syncfusion.com/js/production/api/spreadsheet/open',
    saveUrl:
'https://services.syncfusion.com/js/production/api/spreadsheet/save',

```

```
// Removed the unwanted support for this samples
showFormulaBar: false, showSheetTabs: false, allowInsert: false,
allowDelete: false, allowMerge: false
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [Worksheet](#)
- [Rows and columns](#)

Global local in EJ2 JavaScript Spreadsheet control

Localization

The [Localization](#) library allows you to localize the default text content of the Spreadsheet. The Spreadsheet has static text on some features (cell formatting, Merge, Data validation, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

[locale](#) value and translation object.

The following list of properties and their values are used in the Spreadsheet.

Locale keywords | Text

Cut | Cut

Copy | Copy

Paste | Paste

PasteSpecial | Paste Special

All | All

Values | Values

Formats | Formats

Font | Font

FontSize | Font Size

Bold | Bold

Italic | Italic

Underline | Underline

Strikethrough | Strikethrough

TextColor | Text Color

FillColor | Fill Color

HorizontalAlignment | Horizontal Alignment

AlignLeft | Align Left

AlignCenter | Center

AlignRight | Align Right

VerticalAlignment | Vertical Alignment

AlignTop | Align Top

AlignMiddle | Align Middle

AlignBottom | Align Bottom

InsertFunction | Insert Function

Insert | Insert

Delete | Delete

Rename | Rename

Hide | Hide

Unhide | Unhide

NameBox | Name Box

ShowHeaders | Show Headers

HideHeaders | Hide Headers

ShowGridLines | Show Gridlines

HideGridLines | Hide Gridlines

AddSheet | Add Sheet

ListAllSheets | List All Sheets

FullScreen | Full Screen

CollapseToolbar | Collapse Toolbar

ExpandToolbar | Expand Toolbar

CollapseFormulaBar | Collapse Formula Bar

ExpandFormulaBar | Expand Formula Bar

File | File

Home | Home

Formulas | Formulas

View | View

New | New

Open | Open

SaveAs | Save As

ExcelXlsx | Microsoft Excel

ExcelXls | Microsoft Excel 97-2003

CSV | Comma-separated values

FormulaBar | Formula Bar

Ok | Ok

Close | Close

Cancel | Cancel

Apply | Apply

MoreColors | More Colors

StandardColors | Standard Colors

General | General

Number | Number

Currency | Currency

Accounting | Accounting

ShortDate | Short Date

LongDate | Long Date

Time | Time

Percentage | Percentage

Fraction | Fraction

Scientific | Scientific

Text | Text

NumberFormat | Number Format

MobileFormulaBarPlaceHolder | Enter value or Formula

PasteAlert | You can't paste it here because the copy area and paste area aren't in the same size. Please try pasting in a different range.

DestroyAlert | Are you sure you want to destroy the current workbook without saving and create a new workbook?

SheetRenameInvalidAlert | Sheet name contains invalid character.

SheetRenameEmptyAlert | Sheet name cannot be empty.

SheetRenameAlreadyExistsAlert | Sheet name already exists. Please enter another name.

DeleteSheetAlert | Are you sure you want to delete this sheet?

DeleteSingleLastSheetAlert | A Workbook must contain at least one visible worksheet.

PickACategory | Pick a category

Description | Description

UnsupportedFile | Unsupported File

InvalidUrl | Invalid Url

SUM | Adds a series of numbers and/or cells.

SUMIF | Adds the cells based on the specified condition.

SUMIFS | Adds the cells based on the specified conditions.

ABS | Returns the value of a number without its sign.

RAND | Returns a random number between 0 and 1.

RANDBETWEEN | Returns a random integer based on the specified values.

FLOOR | Rounds a number down to the nearest multiple of a given factor.

CEILING | Rounds a number up to the nearest multiple of a given factor.

PRODUCT | Multiplies a series of numbers and/or cells.

AVERAGE | Calculates average for the series of numbers and/or cells excluding text.

AVERAGEIF | Calculates average for the cells based on the specified criterion.

AVERAGEIFS | Calculates average for the cells based on the specified conditions.

AVERAGEA | Calculates the average for the cells evaluating TRUE as 1 text and FALSE as 0.

COUNT | Counts the cells that contain numeric values in a range.

COUNTIF | Counts the cells based on the specified condition.

COUNTIFS | Counts the cells based on specified conditions.

COUNTA | Counts the cells that contain values in a range.

MIN | Returns the smallest number of the given arguments.

MAX | Returns the largest number of the given arguments.

DATE | Returns the date based on the given year, month, and day.

DAY | Returns the day from the given date.

DAYS | Returns the number of days between two dates.

IF | Returns value based on the given expression.

IFS | Returns value based on the given multiple expressions.

AND | Returns TRUE if all the arguments are TRUE otherwise returns FALSE.

OR | Returns TRUE if any of the arguments are TRUE otherwise returns FALSE.

IFERROR | Returns value if no error found else it will return specified value.

CHOOSE | Returns a value from the list of values based on the index number.

INDEX | Returns the value of the cell in a given range based on row and column number.

FIND | Returns the position of a string within another string which is case sensitive.

CONCATENATE | Combines two or more strings together.

CONCAT | Concatenates a list or a range of text strings.

SUBTOTAL | Returns subtotal for a range using the given function number.

RADIANS | Converts degrees into radians.

MATCH | Returns the relative position of a specified value in the given range.

DefineNameExists | This name already exists try a different name.

CircularReference | When a formula refers to one or more circular references this may result in an incorrect calculation.

ShowRowsWhere | Show rows where |

CustomFilterDatePlaceHolder | Choose a date

CustomFilterPlaceHolder | Enter the value

CustomFilter | Custom Filter

Between | Between

MatchCase | Match Case

DateTimeFilter | DateTime Filters

Undo | Undo

Redo | Redo

DateFilter | Date Filters

TextFilter | Text Filters

NumberFilter | Number Filters

ClearFilter | Clear Filter

NoResult | No Matches Found

FilterFalse | False

FilterTrue | True

Blanks | Blanks

SelectAll | Select All

GreaterThanOrEqual | Greater Than Or Equal

GreaterThan | Greater Than

LessThanOrEqual | Less Than Or Equal

LessThan | Less Than

NotEqual | Not Equal

Equal | Equal

Contains | Contains

EndsWith | Ends With

StartsWith | Starts With

ClearButton | Clear

FilterButton | Filter

CancelButton | Cancel

OKButton | OK

Search | Search

Link | Link

Hyperlink | Hyperlink

EditHyperlink | Edit Hyperlink

OpenHyperlink | Open Hyperlink

RemoveHyperlink | Remove Hyperlink

InsertLink | Insert Link

EditLink | Edit Link

WebPage | WEB PAGE

ThisDocument | THIS DOCUMENT

DisplayText | Display Text

Url | URL

CellReference | Cell Reference

Sheet | Sheet

DefinedNames | Defined Names

EnterTheTextToDisplay | Enter the text to display

EnterTheUrl | Enter the URL

ProtectSheet | Protect Sheet

UnprotectSheet | Unprotect Sheet

SelectCells | Select cells

FormatCells | Format cells

FormatRows | Format rows

Format Columns | Format columns

InsertLinks | Insert links

ProtectContent | Protect the contents of locked cells

ProtectAllowUser | Allow all users of this worksheet to |

EditAlert | The cell you're trying to change is protected. To make a change, unprotect the sheet.

FindReplaceTooltip | Find & Replace

InsertingEmptyValue | Reference value is not valid.

ByRow | By Row

ByColumn | By Column
MatchExactCellElements | Match Exact Cell Contents
EnterCellAddress | Enter Cell Address
FindAndReplace | Find and Replace
ReplaceAllEnd | matches replaced with.
FindNextBtn | Find Next
FindPreviousBtn | Find Previous
ReplaceBtn | Replace
ReplaceAllBtn | Replace All
GotoHeader | Go To
GotoSpecialHeader | Go To Special
SearchWithin | Search within
SearchBy | Search by
Reference | Reference
Workbook | Workbook
NoElements | We couldn't find what you were looking for.
FindWhat | Find what
ReplaceWith | Replace with
EnterValue | Enter Value
DefineNameInvalid | The name that you entered is not valid.
FindValue | Find Value
ReplaceValue | Replace Value
DataValidation | Data Validation
CLEARALL | CLEAR ALL
APPLY | APPLY
CellRange | Cell Range
Allow | Allow
Data | Data
Minimum | Minimum
Maximum | Maximum
IgnoreBlank | Ignore blank
WholeNumber | Whole Number
Decimal | Decimal

Date | Date

TextLength | Text Length

List | List

NotBetween | Not between

EqualTo | Equal to

NotEqualTo | Not equal to

Greaterthan | Greater than

Lessthan | Less than

GreaterThanOrEqaulTo | Greater than or eqaul to

LessThanOrEqualTo | Less than or equal to

InCellDropDown | In-cell-dropdown

Sources | Sources

Value | Value

Retry | Retry

DialogError | The list source must be a reference to a single row or column.

ValidationError | This value doesn't match the data validation restrictions defined for the cell.

HideRow | Hide Row

HideRows | Hide Rows

UnHideRows | UnHide Rows

HideColumn | Hide Column

HideColumns | Hide Columns

UnHideColumns | UnHide Columns

InsertRow | Insert Row

InsertRows | Insert Rows

InsertColumn | Insert Column

InsertColumns | Insert Columns

DeleteRow | Delete Row

DeleteRows | Delete Rows

DeleteColumn | Delete Column

DeleteColumns | Delete Columns

Borders | Borders

TopBorders | Top Borders

LeftBorders | Left Borders

RightBorders | Right Borders

BottomBorders | Bottom Borders

AllBorders | All Borders

HorizontalBorders | Horizontal Borders

VerticalBorders | Vertical Borders

OutsideBorders | Outside Borders

InsideBorders | Inside Borders

NoBorders | No Borders

BorderColor | Border Color

BorderStyle | Border Style

INTERCEPT | Calculates the point of the Y-intercept line via linear regression.

SLOPE | Returns the slope of the line from linear regression of the data points.

FreezePanes | Freeze Panes

FreezeRows | Freeze Rows

FreezeColumns | Freeze Columns

UnfreezePanes | Unfreeze Panes

UnfreezeRows | Unfreeze Rows

UnfreezeColumns | Unfreeze Columns

MergeCells | Merge Cells

MergeAll | Merge All

MergeHorizontally | Merge Horizontally

MergeVertically | Merge Vertically

Unmerge | Unmerge

UnmergeCells | Unmerge Cells

SortAscending | Ascending

SortDescending | Descending

CustomSort | Custom Sort

ClearAllFilter | Clear

ReapplyFilter | Reapply

Clear | Clear

ClearContents | Clear Contents

ClearAll | Clear All

ClearFormats | Clear Formats

[ClearHyperlinks](#) | [Clear Hyperlinks](#)

[Image](#) | [Image](#)

[AddColumn](#) | [Add Column](#)

[SortBy](#) | [Sort by](#)

[ThenBy](#) | [Then by](#)

[Chart](#) | [Chart](#)

[Column](#) | [Column](#)

[Bar](#) | [Bar](#)

[Area](#) | [Area](#)

[Pie](#) | [Pie](#)

[Doughnut](#) | [Doughnut](#)

[PieAndDoughnut](#) | [Pie/Doughnut](#)

[Line](#) | [Line](#)

[Radar](#) | [Radar](#)

[Scatter](#) | [Scatter](#)

[ChartDesign](#) | [Chart Design](#)

[ClusteredColumn](#) | [Clustered Column](#)

[StackedColumn](#) | [Stacked Column](#)

[StackedColumn100](#) | [100% Stacked Column](#)

[ClusteredBar](#) | [Clustered Bar](#)

[StackedBar](#) | [Stacked Bar](#)

[StackedBar100](#) | [100% Stacked Bar](#)

[StackedArea](#) | [Stacked Area](#)

[StackedArea100](#) | [100% Stacked Area](#)

[StackedLine](#) | [Stacked Line](#)

[StackedLine100](#) | [100% Stacked Line](#)

[AddChartElement](#) | [Add Chart Element](#)

[Axes](#) | [Axes](#)

[AxisTitle](#) | [Axis Title](#)

[ChartTitle](#) | [Chart Title](#)

[DataLabels](#) | [Data Labels](#)

[Gridlines](#) | [Gridlines](#)

[Legends](#) | [Legends](#)

PrimaryHorizontal | Primary Horizontal

PrimaryVertical | Primary Vertical

None | None

AboveChart | Above Chart

Center | Center

InsideEnd | Inside End

InsideBase | Inside Base

OutsideEnd | OutSide End

PrimaryMajorHorizontal | Primary Major Horizontal

PrimaryMajorVertical | Primary Major Vertical

PrimaryMinorHorizontal | Primary Minor Horizontal

PrimaryMinorVertical | Primary Minor Vertical

Right | Right

Left | Left

Bottom | Bottom

Top | Top

SwitchRowColumn | Switch Row/Column

ChartTheme | Chart Theme

ChartType | Chart Type

Material | Material

Fabric | Fabric

Bootstrap | Bootstrap

HighContrastLight | HighContrastLight

MaterialDark | MaterialDark

FabricDark | FabricDark

HighContrast | HighContrast

BootstrapDark | BootstrapDark

Bootstrap4 | Bootstrap4

VerticalAxisTitle | Vertical Axis Title

HorizontalAxisTitle | Horizontal Axis Title

EnterTitle | Enter Title

ProtectWorkbook | Protect Workbook

Password | Password (optional) |

unProtectPassword | Password

EnterThePassword | Enter the password

ConfirmPassword | Confirm Password

EnterTheConfirmPassword | Re-enter your password

PasswordAlert | Confirmation password is not identical

UnProtectWorkbook | Unprotect Workbook

UnProtectPasswordAlert | The password you supplied is not correct.

InCorrectPassword | Unable to open the file or worksheet with the given password.

PasswordAlertMsg | Please enter the password

ConfirmPasswordAlertMsg | Please enter the confirm password

IsProtected | is protected

Loading translations

To load translation object in an application, use [load](#) function of the [L10n](#) class.

The following example demonstrates the Spreadsheet in **French** culture. In the below sample we have translated the ribbon tab names and Home tab content (clipboard, cell style).

INDEX.TS

```
import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from './datasource.ts';
import { L10n, enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
L10n.load({
  'fr-CH': {
    'spreadsheet': {
      'File': 'Fichier',
      'Home': 'Accueil',
      'Insert': 'Insérer',
      'Formulas': 'Formules',
      'Data': 'Les données',
      'View': 'Vue',
      'Cut': 'Coupe',
      'Copy': 'Copie',
      'Paste': 'Pâte',
      'PasteSpecial': 'Pâte spéciale',
      'All': 'Tous les',
      'Values': 'Valeurs',
      'Formats': 'Les formats',
      'Font': 'fonte',
      'FontSize': 'Taille de police',
      'Bold': 'Audacieux',
      'Italic': 'Italique',
      'Underline': 'Souligner',
      'Strikethrough': 'Barré',
      'TextColor': 'Couleur du texte',
      'FillColor': 'La couleur de remplissage',
      'HorizontalAlignment': 'Alignement horizontal',
      'AlignLeft': 'Alignez à gauche',
    }
  }
});
```

```

        'AlignCenter': 'centre',
        'AlignRight': 'Aligner à droite',
        'VerticalAlignment': 'Alignement vertical',
        'AlignTop': 'Aligner en haut',
        'AlignMiddle': 'Aligner le milieu',
        'AlignBottom': 'Aligner le bas',
        'InsertFunction': 'Insérer une fonction',
        'Delete': 'Effacer',
        'Rename': 'Rebaptiser',
        'Hide': 'Cacher',
        'Unhide': 'Démasquer',
        'NumberFormat': 'Nombre Format',
    }
}
});
let columns: ColumnModel[] = [{ width: 100 }, { width: 130 }, { width: 96 },
    { width: 130 }, { width: 130 }, { width: 96 },
    { width: 100 }, { width: 100 }, { width: 110 }, { width: 100 }, { width:
130 }, { width: 150 }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    // Specifies the locale
    locale: 'fr-CH',
    sheets: [{ ranges: [{ dataSource: defaultData }], columns: columns
    }],
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
        <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Internationalization

The Internationalization library is used to globalize number, date, and time values in the spreadsheet component.

The following example demonstrates the Spreadsheet in French [fr-CH] culture. In the below sample we have globalized the Date(Date column), Time(Time column), and Currency(Amount column) formats.

INDEX.TS

```

import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from './datasource.ts';
import { L10n, enableRipple } from '@syncfusion/ej2-base';
import { loadCldr, L10n, setCulture, setCurrencyCode } from
 '@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as cagregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
enableRipple(true);
loadCldr(currencies, cagregorian, numbers, timeZoneNames, numberingSystems);
setCulture('fr-CH');
setCurrencyCode('EUR');
L10n.load({
    'fr-CH': {
        'spreadsheet': {
            'File': 'Fichier',

```

```

        'Home': 'Accueil',
        'Insert': 'Insérer',
        'Formulas': 'Formules',
        'Data': 'Les données',
        'View': 'Vue',
        'Cut': 'Coupe',
        'Copy': 'Copie',
        'Paste': 'Pâte',
        'PasteSpecial': 'Pâte spéciale',
        'All': 'Tous les',
        'Values': 'Valeurs',
        'Formats': 'Les formats',
        'Font': 'fonte',
        'FontSize': 'Taille de police',
        'Bold': 'Audacieux',
        'Italic': 'Italique',
        'Underline': 'Souligner',
        'Strikethrough': 'Barré',
        'TextColor': 'Couleur du texte',
        'FillColor': 'La couleur de remplissage',
        'HorizontalAlignment': 'Alignement horizontal',
        'AlignLeft': 'Alignez à gauche',
        'AlignCenter': 'centre',
        'AlignRight': 'Aligner à droite',
        'VerticalAlignment': 'Alignement vertical',
        'AlignTop': 'Aligner en haut',
        'AlignMiddle': 'Aligner le milieu',
        'AlignBottom': 'Aligner le bas',
        'InsertFunction': 'Insérer une fonction',
        'Delete': 'Effacer',
        'Rename': 'Rebaptiser',
        'Hide': 'Cacher',
        'Unhide': 'Démasquer',
        'NumberFormat': 'Nombre Format',
    }
}
});
let columns: ColumnModel[] = [{ width: 100 }, { width: 130 }, { width: 96 },
    { width: 130 }, { width: 130 }, { width: 96 },
    { width: 100 }, { width: 100 }, { width: 110 }, { width: 100 }, { width:
130 }, { width: 150 }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    // Specifies the locale
    locale: 'fr-CH',
    sheets: [{ ranges: [{ dataSource: defaultData }], columns: columns
    }],
    created: (): void => {
        //Applies cell and number formatting to specified range of the
        active sheet
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:F1');
        spreadsheet.numberFormat('$#,##0.00', 'F2:F11');
    }
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Grid</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Grid Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Spreadsheet component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Spreadsheet, set the [enableRtl](#) to true.

INDEX.TS

```
import { Spreadsheet, SheetModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from './datasource.ts';
import { L10n, enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
L10n.load({
  'ar-AE': {
    'spreadsheet': {
      "File": "ملف",
      "Home": "هم",
      "Insert": "إدراج",
      "Formulas": "الصيغ",
      "View": "معاينة",
      "Data": "البيانات",
      "Cut": "قطع",
      "Copy": "نسخ",
      "Paste": "معجون",
      "PasteSpecial": "لصق خاص",
      "All": "جميع",
      "Values": "القيم",
      "Formats": "شكل",
      "Font": "الخط",
      "FontSize": "حجم الخط",
      "Bold": "جريء",
      "Italic": "مائل",
      "Underline": "أكد",
      "Strikethrough": "يتوسطه",
      "TextColor": "لون الخط",
      "FillColor": "لون التعبئة",
      "HorizontalAlignment": "المحاذاة الأفقية",
      "AlignLeft": "محاذاة إلى اليسار",
      "AlignCenter": "مركز",
      "AlignRight": "محاذاة إلى اليمين",
      "VerticalAlignment": "محاذاة عمودية",
      "AlignTop": "محاذاة أعلى",
      "AlignMiddle": "محاذاة الأوسط",
      "AlignBottom": "أسفل محاذاة",
      "InsertFunction": "إدراج وظيفة",
      "Delete": "حذف",
      "Rename": "إعادة تسمية",
      "Hide": "إخفاء",
      "Unhide": "ظهار"
    }
  }
});
let columns: ColumnModel[] = [{ width: 100 }, { width: 130 }, { width: 96 },
  { width: 130 }, { width: 130 }, { width: 96 },
  { width: 100 }, { width: 100 }, { width: 110 }, { width: 100 }, { width:
  130 }, { width: 150}];
let spreadsheet: Spreadsheet = new Spreadsheet({
  locale: 'ar-AE',
```

```

        enableRtl: true,
        sheets: [{ ranges: [{ dataSource: defaultData }], columns: columns
    }]
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Localization](#)

Undo redo in EJ2 JavaScript Spreadsheet control

Undo option helps you to undone the last action performed and **Redo** option helps you to do the same action which is reverted in the Spreadsheet. You can use the [allowUndoRedo](#) property to enable or disable undo redo functionality in spreadsheet.

* The default value for **allowUndoRedo** property is **true**.

By default, the **UndoRedo** module is injected internally into Spreadsheet to perform undo redo.

Undo

It reverses the last action you performed with Spreadsheet. Undo can be done by any of the following ways:

- Select the undo item from HOME tab in Ribbon toolbar.
- Use **Ctrl + Z** keyboard shortcut to perform the undo.
- Use the [undo](#) method programmatically.

Redo

It reverses the last undo action you performed with Spreadsheet. Redo can be done by any of the following ways:

- Select the redo item from HOME tab in Ribbon toolbar.
- Use **Ctrl + Y** keyboard shortcut to perform the redo.
- Use the [redo](#) method programmatically.

Update custom actions in UndoRedo collection

You can update your own custom actions in UndoRedo collection, by using the [updateUndoRedoCollection](#) method. And also customize the undo redo operations of your custom action by using **actionComplete** event.

The following code example shows **How to update and customize your own actions for undo redo** functionality in the Spreadsheet control.

INDEX.TS

```

import { Spreadsheet, getRangeIndexes, ColumnModel } from '@syncfusion/ej2-spreadsheet';
import { defaultData } from './datasource.ts';
import { enableRipple, addClass, removeClass } from '@syncfusion/ej2-base';

```



```

let columns: ColumnModel[] = [
    {
        width: 130
    },
    {
        width: 92
    },
    {
        width: 96
    }
];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ ranges: [{ dataSource: defaultData }], columns: columns }],
    actionComplete: (args): void => {
        let actionEvents: any = args;
        if (actionEvents.eventArgs.action == "customCSS") {
            let Element:HTMLElement =
spreadsheet.getCell(actionEvents.eventArgs.rowIdx,actionEvents.eventArgs.col
Idx);

            if (actionEvents.eventArgs.requestType == "undo") {
                removeClass([Element], 'customClass'); // To remove the
custom class in undo action
            }
            else {
                addClass([Element], 'customClass'); // To add the custom class
in redo action
            }
        }
    }
});
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
document.getElementById("customBtn").addEventListener('click',
updateCollection);
function updateCollection() {
    let cell: string = spreadsheet.getActiveSheet().activeCell;
    let cellIdx: number[] = getRangeIndexes(cell);
    let Element: HTMLElement = spreadsheet.getCell(cellIdx[0], cellIdx[1]);
    if (!Element.classList.contains("customClass")) {
        addClass([Element], 'customClass'); // To add the custom class in
active cell element
        spreadsheet.updateUndoRedoCollection({ eventArgs: { class:
'customClass', rowIdx: cellIdx[0], colIdx: cellIdx[1], action: 'customCSS' }
}); // To update the undo redo collection
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">

```

```

    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <style>
        .customClass.e-cell {
            background-color: red;
        }
    </style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->
    <button id="customBtn" class="e-btn"> add/remove Class</button>

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Sorting](#)
- [Filtering](#)
- [Hyperlink](#)

Accessibility in EJ2 JavaScript Spreadsheet control

The Spreadsheet control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Spreadsheet control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

WAI-ARIA attributes

The Spreadsheet control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Spreadsheet control:

Attributes	Purpose
-----	-----
grid (role)	This role is added to the spreadsheet content table and describes the collection of rows and columns.
gridcell (role)	This role is added to the cell element and describes the rows <td> element.
rowheader (role)	This role is added to the row header and describes the header of the rows.
colheader (role)	This role is added to the column header and describes the header of the columns.
aria-rowindex (attribute)	This attribute describes the table's row index in the spreadsheet.
aria-colindex (attribute)	This attribute describes the table's column index in the spreadsheet.
aria-selected (attribute)	This attribute describes an item's (cell, menu, checkbox, etc.) current selected state in the spreadsheet.
aria-rowcount (attribute)	This attribute describes the total number of rows in the table.
aria-colcount (attribute)	This attribute describes the total number of columns in the table.
aria-busy (attribute)	This attribute describes a currently updated or modified element.
aria-label (attribute)	This attribute describes the accessible name for the interactive elements.
textbox (role)	This role is assigned to the textbox that accepts text input.
menu (role)	This role has been added to the menu and describes the menu items.
aria-expanded (attribute)	This attribute describes the control (for example, dropdown) is expanded or collapsed.
aria-multiline (attribute)	This attribute defines what the Alt + Enter key does in the spreadsheet editor.

Keyboard interaction

The Spreadsheet control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Spreadsheet control.

Press	To do this
-----	-----
Up arrow	Navigate from the active cell to the previous cell in the same column.

| Down arrow | Navigate from the active cell to the next cell in the same column. |

| Left arrow | Navigate from the active cell to the previous cell in the same row. |

| Right arrow | Navigate from the active cell to the next cell in the same row. |

| Tab | Navigate the active cell to the next cell in the same row. |

| Shift + Tab | Navigate the active cell to the previous cell in the same row. |

| Home | Moves the selection to starting column in worksheet. |

| Ctrl + Home | Move the selection to the first visible cell on a worksheet. |

| Shift + Home | Extend the cell selection to the first column of a worksheet. |

| Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |

| Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |

| Page Up | Move page up. |

| Page Down | Move page down. |

| Shift + Page Up | Perform page up by selecting all cells between. |

| Shift + Page Down | Perform page down by selecting all cells between. |

| Ctrl + Up | Navigate to the non-blank cell before the active cell in the same column. |

| Ctrl + Down | Navigate to the last non-blank cell in the same column as the active cell. |

| Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |

| Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |

| Shift + Up | Extend the selection of cells to the previous row. |

| Shift + Down | Extend the selection of cells to the next row. |

| Shift + Left | Extend the selection of cells to the previous column. |

| Shift + Right | Extend the selection of cells to the next column. |

| Ctrl + Shift + Up | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |

| Ctrl + Shift + Down | Extend the cell selection to the last non-blank cell in the same column as the active cell. |

| Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |

| Enter | Navigate the active cell to the next cell in the same column. |

| Shift + Enter | Navigate to the previous cell in the same column from the active cell. |

| Alt + Enter | While editing, add a new line. |

| Enter | Complete the cell editing and select the cell below in the same column. |

- | Shift + Enter | Complete the cell editing and select the cell above in the same column. |
- | Tab | Complete the cell editing and select the next cell in the same row. |
- | Shift + Tab | Complete the cell editing and select the previous cell in the same row. |
- | Alt | Focus on the active ribbon tab. |
- | Left | Move the focus to the previous items in the ribbon content. |
- | Right | Move the focus to the next items in the ribbon content. |
- | Alt + Down | Open the ribbon dropdown menu. |
- | Esc / Alt + Up | Close the ribbon dropdown menu. |

Ensuring accessibility

The Spreadsheet control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Spreadsheet control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Spreadsheet control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 controls](#)

Styles in EJ2 JavaScript Spreadsheet control

To modify the Spreadsheet appearance, you need to override the default CSS of the spreadsheet. Please find the CSS structure that can be used to modify the Spreadsheet appearance. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Customizing the Spreadsheet

Use the below CSS to customize the Spreadsheet root element.

```
,  
  
.e-spreadsheet {  
font-family: cursive;  
}  
,
```

Header

Customizing the Spreadsheet Ribbon

Use the below CSS to customize the Spreadsheet Ribbon.

```
,  
  
.e-spreadsheet .e-ribbon {  
background-color: #808080;  
}  
,
```

Customizing the Spreadsheet formula bar panel

You can customize the Spreadsheet formula bar panel by using this CSS.

,

```
.e-spreadsheet .e-formula-bar-panel {  
border: none;  
}
```

,

Customizing the Spreadsheet formula bar text

You can customize the Spreadsheet formula bar text by using this CSS.

,

```
.e-spreadsheet .e-formula-bar-panel .e-formula-bar {  
font-weight: bold;  
}
```

,

*Sheet**Customizing the Spreadsheet sheet element*

Using this CSS, you can customize the Spreadsheet sheet element.

,

```
.e-spreadsheet .e-sheet-panel .e-sheet {  
border-color: #000000;  
}
```

,

Customizing the Spreadsheet sheet header

Use the below CSS to customize the Spreadsheet sheet header.

,

```
.e-spreadsheet .e-sheet-panel .e-sheet .e-header-panel {  
font-style: oblique;  
}
```

,

Customizing the Spreadsheet row header

Use the below CSS to customize the Spreadsheet row header.

,

```
.e-spreadsheet .e-row-header .e-header-cell {  
color: #0000FF !important;  
}
```

,

Customizing the Spreadsheet column header

Use the below CSS to customize the Spreadsheet column header.

,

```
.e-spreadsheet .e-column-header .e-header-cell {  
color: #0000FF !important;  
}
```

,

Customizing the Spreadsheet selection element

Customize the Spreadsheet selection element.

,

```
.e-spreadsheet .e-selection {  
border-color: #0000FF;  
}
```

,

Customizing the Spreadsheet active cell element

Customize the Spreadsheet active cell element.

,

```
.e-spreadsheet .e-active-cell {  
border-color: #0000FF;  
}
```

,

Customizing the Spreadsheet cell element

Using this CSS, you can customize the Spreadsheet cell element.

,

```
.e-spreadsheet .e-cell {  
background-color: #0078d7 !important;  
}
```

,

Ribbon Items

Customizing the Spreadsheet sorting icon

Use the below CSS to customize the Spreadsheet sorting icon in the Spreadsheet ribbon. You can use the available Syncfusion [icons](#) based on your theme.

,

```
.e-spreadsheet .e-sort-filter-icon:before {
```



```
background-color: #deecf9;
content: '\e306';
}
`
```

Customizing the filter dialog content

Use the below CSS to customize the Spreadsheet filter dialog content element.

```
`
.e-spreadsheet .e-filter-popup .e-dlg-content {
background-color: #deecf9;
}
`
```

Customizing the filter dialog footer

Spreadsheet filter dialog footer element can be customized by using the below CSS.

```
`
.e-spreadsheet .e-filter-popup .e-footer-content {
background-color: #ccffff;
}
`
```

Customizing the filter dialog input element

Use the below CSS to customize the Spreadsheet filter dialog input element.

```
`
.e-spreadsheet .e-filter-popup .e-input-group input.e-input{
font-family: cursive;
}
`
```

Customizing the filter dialog button element

Use the below CSS to customize the Spreadsheet filter dialog button element.

```
`
.e-spreadsheet .e-filter-popup .e-btn{
font-family: cursive;
}
`
```

Customizing the Excel filter dialog number filters element

Spreadsheet Excel filter dialog number filters element can be customized by using the below CSS.

```
`
```

```
.e-spreadsheet .e-filter-popup .e-contextmenu-wrapper ul{
background-color: #deecf9;
}
`
```

Footer

Customizing the Spreadsheet sheet tab panel

Spreadsheet sheet tab panel can be customized by using the below CSS.

```
.e-spreadsheet .e-sheet-tab-panel {
background: #08fbfb;
}
`
```

Customizing the Spreadsheet sheet tab

Spreadsheet sheet tab element can be customized by using the below CSS.

```
.e-spreadsheet .e-sheet-tab-panel .e-tab-header .e-toolbar-item.e-active .e-tab-wrap .e-tab-text {
font-weight: bold;
}
`
```

Use Cases

Collaborative editing in EJ2 JavaScript Spreadsheet control

The collaborative editing support allows you to work at a spreadsheet collaboratively with other users. Multiple users can access to the the same spreadsheet simultaneously.

Dependencies

The following list of dependencies are required to use the collaborative editing support in spreadsheet.

```
`js
|-- @aspnet/signalr
|-- eventsourcing
|-- request
|-- ws
`
```

* Run the command `npm i @aspnet/signalr` to install `@aspnet/signalr` packages in your application.

Client configuration

To broadcast the data for every action in the spreadsheet, you need to transfer the data to the server through `send` method in `actionComplete` event and receive the same data by using the `dataReceived`

method. In the `dataReceived` method, you need to update the action to the connected clients through `updateAction` method.

The following code example shows `Collaborative Editing` support in the Spreadsheet control.

```
`ts
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import * as signalR from '@aspnet/signalr';
import { data } from './datasource.ts';
// For signalR Hub connection
const connection: signalR.HubConnection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost
from AspNetCore service
skipNegotiation: true,
transport: signalR.HttpTransportType.WebSockets
}).build();
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
sheets: [{
ranges: [{ dataSource: data }],
columns: [{ width: 130 }, { width: 110 }, { width: 110 },
{ width: 90 }, { width: 90 }, { width: 90 }, { width: 90 }, { width: 90 }
}],
actionComplete: (args) => {
connection.send('BroadcastData', JSON.stringify(args)); // send the action data to the server
},
});
spreadsheet.appendTo('#spreadsheet');
connection.on('dataReceived', (data: string) => {
let model: CollaborativeEditArgs = JSON.parse(data) as CollaborativeEditArgs;
spreadsheet.updateAction(model); // update the action to the connected clients
});
connection.start().then(() => { // to start the server
console.log('server connected!!!');
}).catch(err => console.log(err));
`
```

Server configuration

To make the communication between the server to the connected clients and from clients to the server, you need to configure the signalR Hubs using the following code.

```
`ts
import * as signalR from '@aspnet/signalr';
// For signalR Hub connection
const connection: signalR.HubConnection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost
from AspNetCore service
skipNegotiation: true,
transport: signalR.HttpTransportType.WebSockets
}).build();
`
```

Hub configuration

Initially create a ASP.NET Core project and add the hub file for sending and receiving the data between server and clients.

```
`ts
using Microsoft.AspNetCore.SignalR;
using System.Threading.Tasks;
namespace WebApplication.Hubs
{
public class SpreadsheetHub : Hub
{
public async Task BroadcastData(string data)
{
await Clients.Others.SendAsync("dataReceived", data);
}
}
}
`
```

To configure the SignalR middleware by registering the following service in the `ConfigureServices` method of the `Startup` class.

```
`ts
services.AddSignalR(e =>
{
```

e.MaximumReceiveMessageSize = int.MaxValue; // Option to increase message size for inserting image feature. By default, SignalR send messages up to 32 KB.

```
});
`
```

To set up the SignalR routes by calling MapHub in the `Configure` method of the `Startup` class.

```
`ts
app.UseEndpoints(endpoints =>
{
    endpoints.MapRazorPages();
    endpoints.MapHub<SpreadsheetHub>("/hubs/spreadsheethub");
});
`
```

For hosting the service, you may use the above code snippet or download and run the [local service](#).

Prevent the particular action update for collaborative client

Using the `action` argument from the `actionComplete` event, you can prevent the particular action update for collaborative client.

The following code example shows how to prevent collaborative client from updating the `format` action.

```
`ts
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import * as signalR from '@aspnet/signalr';
import { data } from './datasource.ts';
// For signalR Hub connection
const connection: signalR.HubConnection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost
from AspNetCore service
skipNegotiation: true,
transport: signalR.HttpTransportType.WebSockets
}).build();
//Initialize the SpreadSheet control
let spreadsheet: Spreadsheet = new Spreadsheet({
sheets: [{
ranges: [{ dataSource: data }],
columns: [{ width: 130 }, { width: 110 }, { width: 110},
{ width: 90 }, { width: 90 }, {width: 90}, { width: 90 }, {width: 90}]
}],

```

```

actionComplete: (args) => {
  if (args.action !== 'format') { // prevent the format action
    connection.send('BroadcastData', JSON.stringify(args)); // send the action data to the server
  }
},
});
spreadsheet.appendTo('#spreadsheet');
connection.on('dataReceived', (data: string) => {
  let model: CollaborativeEditArgs = JSON.parse(data) as CollaborativeEditArgs;
  spreadsheet.updateAction(model); // update the action to the connected clients
});
connection.start().then(() => { // to start the server
  console.log('server connected!!!');
}).catch(err => console.log(err));
`

```

Perform import action for collaborative clients

Using the `action` argument from the [actionComplete](#) event, you can identify whether the import action is performed or not. If the action is `import`, then you need to send the `response data` to the server and also update the same to the collaborative clients.

The following code example shows how to perform the import functionality for collaborative clients.

```

`ts
import { Spreadsheet, CollaborativeEditArgs } from '@syncfusion/ej2-spreadsheet';
import * as signalR from '@microsoft/signalr';
import { isNullOrUndefined } from "@syncfusion/ej2-base";
// For signalR Hub connection.
const connection: signalR.HubConnection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', {
  skipNegotiation: true,
  transport: signalR.HttpTransportType.WebSockets
}).build();
let spreadsheet: Spreadsheet = new Spreadsheet({
  openUrl: 'https://services.syncfusion.com/js/production/api/spreadsheet/open',
  actionComplete: (args: {action: string, response: any}) => {
    if (args.action === 'import') {

```

```
// Send the action data to the server in args.response at the time of importing an excel file.
connection.send("BroadcastData", JSON.stringify(args.response.data));
}
else {
// Send the action data to the server for other than import actions.
connection.send("BroadcastData", JSON.stringify(args));
}
},
});
spreadsheet.appendTo('#spreadsheet');
connection.on('dataReceived', (data: string) => {
const model: CollaborativeEditArgs = JSON.parse(
data
) as CollaborativeEditArgs;
// Condition to check whether action performed is import.
if (isNullOrUndefined(model['action'])) {
// Load the imported excel file data as JSON to the connected clients.
const jsonData: object = { Workbook: model };
spreadsheet.openFromJson({ file: jsonData });
}
else {
// Update the action details to the connected clients.
spreadsheet.updateAction(model);
}
});
connection
.start()
.then(() => {
console.log('server connected!!!');
})
.catch(err => console.log(err));
,
```

[See Also](#)

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

How To

Sort a range by custom list in EJ2 JavaScript Spreadsheet control

You can also define the sorting of cell values based on your own customized personal list. In this article, custom list is achieved using custom sort comparer.

In the following demo, the Trustworthiness column is sorted based on the custom lists Perfect, Sufficient, and Insufficient.

INDEX.TS

```
import { Spreadsheet, CellModel } from '@syncfusion/ej2-spreadsheet';
import { DataManager, DataUtil } from '@syncfusion/ej2-data';
import { tradeData } from './datasource.ts';
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ ranges: [{ dataSource: new DataManager(tradeData) }] }],
    dataBound: function () {
        if (spreadsheet.activeSheetIndex === 0) {
            spreadsheet.sort({sortDescriptors: { field: 'F', sortComparer:
mySortComparer }, containsHeader: true}, 'A1:H20');
        }
    },
    sortComplete: function (args) {
        spreadsheet.selectRange(args.range);
        // code here.
    }
});
// custom sort comparer to sort based on the custom list.
let customList: string[] = ['Perfect', 'Sufficient', 'Insufficient'];
function mySortComparer(x: CellModel, y: CellModel): number {
    let comparer: Function = DataUtil.fnSort('Ascending');
    return comparer(x ? customList.indexOf(x.value) : x, y ?
customList.indexOf(y.value) : y);
};
//Render the initialized Spreadsheet
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
```



```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->

  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[See Also](#)

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

Create a object structure in EJ2 JavaScript Spreadsheet control

This topic guides you to construct a JSON structure that can be passed to the [openFromJson](#) method to render the spreadsheet. The JSON structure is an object with the key as **Workbook** and the [properties](#) of the spreadsheet as value.

```
`js
{ Workbook: {} }
`
```

The following properties are the root level properties of the **Workbook** object.

Property	Type	Description
activeSheetIndex	number	Specifies active sheet index in the workbook.
sheets	Sheet[]	Contains a list of sheet properties.
definedNames	DefineName[]	Specifies the name for a range and uses it in the formula for calculation.

The following table defines each property of the **Sheet**.

Property	Type	Description
name	string	Specifies the name of the sheet.
selectedRange	string	Specifies selected range in the sheet.
activeCell	string	Specifies active cell within selectedRange in the sheet.
topLeftCell	string	Specified cell will be positioned at the upper-left corner of the sheet.
showHeaders	boolean	Specifies to show or hide column and row headers in the sheet.
showGridLines	boolean	Specifies to show or hide gridlines in the sheet.
isProtected	boolean	Specifies to protect the cells in the sheet.
state	SheetState	Specifies the sheet visibility state. There must be at least one visible sheet in Spreadsheet.
columns	Column[]	Contains a list of column properties
rows	Row[]	Contains a list of row properties
protectSettings	ProtectSettings	Configures protect and its options.
conditionalFormats	ConditionalFormat[]	Specifies the conditional formatting for the sheet.

The following table defines each property of the **Column**.

Property	Type	Description
width	number	Specifies the width of the column.

| customWidth | boolean | Specifies custom width of the column. |
 | hidden | boolean | To hide or show the column in the sheet. |

The following table defines each property of the **Row**.

Property	Type	Description
height	number	Specifies the height of the row.
customHeight	boolean	Specifies the custom height of the row.
hidden	boolean	To hide or show the row in the sheet.
cells	Cell[]	Contains a list of cell properties

The following table defines each property of the **Cell**.

Property	Type	Description
value	string	Defines the value of the cell which can be text or number.
formula	string	Defines the formula or expression of the cell.
format	string	Specifies the number format code to display the value in specified number format.
hyperlink	string	Specifies the hyperlink of the cell.
wrap	boolean	Wraps the cell text to the next line, if the text width exceeds the column width.
isLocked	boolean	Specifies the cell whether it is locked or not, for allowing edit range in the spreadsheet protect option.
colSpan	number	Specifies the column-wise cell merge count.
rowSpan	number	Specifies the row-wise cell merge count.
style	CellStyle	Specifies the cell style options.
validation	Validation	Specifies the validation of the cell.
image	Image[]	Specifies the image of the cell.

The following table defines each property of the **CellStyle**.

Property	Type	Description
fontFamily	FontFamily	Specifies font family of the cell.
verticalAlign	VerticalAlign	Specifies vertical align of the cell.
textAlign	TextAlign	Specifies text align style of the cell.
textIndent	string	Specifies text indent style of the cell.
color	string	Specifies font color of the cell.
backgroundColor	string	Specifies the background color of the cell.

fontWeight	**FontWeight**	Specifies font weight of the cell.
fontStyle	**FontStyle**	Specifies font style of the cell.
fontSize	string	Specifies font size of the cell.
textDecoration	**TextDecoration**	Specifies text decoration of the cell.
border	string	Specifies border of the cell.
borderTop	string	Specifies top border of the cell.
borderBottom	string	Specifies bottom border of the cell.
borderLeft	string	Specifies left border of the cell.
borderRight	string	Specifies right border of the cell.

```
`js
```

```
type FontFamily = 'Arial' | 'Arial Black' | 'Axettac Demo' | 'Batang' | 'Book Antiqua' | 'Calibri' | 'Courier' |  

  'Courier New' | 'Din Condensed' | 'Georgia' | 'Helvetica' | 'Helvetica New' | 'Roboto' | 'Tahoma' | 'Times  

  New Roman' | 'Verdana';
```

```
type VerticalAlign = 'bottom' | 'middle' | 'top';
```

```
type TextAlign = 'left' | 'center' | 'right';
```

```
type FontWeight = 'bold' | 'normal';
```

```
type FontStyle = 'italic' | 'normal';
```

```
type TextDecoration = 'underline' | 'line-through' | 'underline line-through' | 'none';
```

```
,
```

The following table defines each property of the **Validation**.

Property	Type	Description
-----	-----	-----
type	ValidationType	Specifies Validation Type.
operator	ValidationOperator	Specifies Validation Operator.
value1	string	Specifies Validation Minimum Value.
value2	string	Specifies Validation Maximum Value.
ignoreBlank	boolean	Specifies IgnoreBlank option in Data Validation.
inCellDropDown	boolean	Specifies InCellDropDown option in Data Validation.
isHighlighted	boolean	Specifies to allow Highlight Invalid Data.

```
`js
```

```
type ValidationType = 'WholeNumber' | 'Decimal' | 'Date' | 'TextLength' | 'List' | 'Time';
```

```
type ValidationOperator = 'Between' | 'NotBetween' | 'EqualTo' | 'NotEqualTo' | 'LessThan' |  

  'GreaterThan' | 'GreaterThanOrEqualTo' | 'LessThanOrEqualTo';
```

The following table defines each property of the `Image`.

Property	Type	Description
src	string	Specifies the image source.
id	string	Specifies image element id.
height	number	Specifies the height of the image.
width	number	Specifies the width of the image.
top	number	Specifies the top position of the image.
left	number	Specifies the left position of the image.

The following table defines each property of the `ConditionalFormat`.

Property	Type	Description
type	<code>HighlightCell</code> or <code>TopBottom</code> or <code>DataBar</code> or <code>ColorScale</code> or <code>IconSet</code>	Specifies Conditional formatting Type.
format	<code>Format</code>	Specifies format.
cFColor	<code>CFCOLOR</code>	Specifies Conditional formatting Highlight Color.
value	string	Specifies Conditional formatting value.
range	string	Specifies Conditional formatting range.

```
`js
type HighlightCell = 'GreaterThan' | 'LessThan' | 'Between' | 'EqualTo' | 'ContainsText' | 'DateOccur' |
'Duplicate' | 'Unique';

type TopBottom = 'Top10Items' | 'Bottom10Items' | 'Top10Percentage' | 'Bottom10Percentage' |
'BelowAverage' | 'AboveAverage';

type DataBar = 'BlueDataBar' | 'GreenDataBar' | 'RedDataBar' | 'OrangeDataBar' | 'LightBlueDataBar' |
'PurpleDataBar';

type ColorScale = 'GYRColorScale' | 'RYGColorScale' | 'GWRCColorScale' | 'RWGColorScale' |
'BWRCColorScale' | 'RWBCColorScale' | 'WRCColorScale' | 'RWColorScale' | 'GWColorScale' |
'WGColorScale' | 'GYColorScale' | 'YGColorScale';

type IconSet = 'ThreeArrows' | 'ThreeArrowsGray' | 'FourArrowsGray' | 'FourArrows' | 'FiveArrowsGray' |
'FiveArrows' | 'ThreeTrafficLights1' | 'ThreeTrafficLights2' | 'ThreeSigns' | 'FourTrafficLights' |
'FourRedToBlack' | 'ThreeSymbols' | 'ThreeSymbols2' | 'ThreeFlags' | 'FourRating' | 'FiveQuarters' |
'FiveRating' | 'ThreeTriangles' | 'ThreeStars' | 'FiveBoxes';

type CFColor = 'RedFT' | 'YellowFT' | 'GreenFT' | 'RedF' | 'RedT';
```

The following table defines each property of the `Format`.

Property	Type	Description
format	string	Specifies the number format code to display the value in specified number format.
style	CellStyle	Specifies the cell style options.

The following table defines each property of the `DefinedName`.

Property	Type	Description
name	string	Specifies a name for the defined name, which can be used in the formula.
scope	string	Specifies scope for the defined name.
comment	string	Specifies comment for the defined name.
refersTo	string	Specifies reference for the defined name.

In the following demo, the JSON structure is passed to the `openFromJson` method to render the spreadsheet in the `created` event.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { jsonData } from './datasource.ts';
let spreadsheet: Spreadsheet = new Spreadsheet({
    created: function () {
        spreadsheet.openFromJson({ file: jsonData });
    }
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <!--Element which is going to render-->

    <div id="container">
    <div id="spreadsheet"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Print in EJ2 JavaScript Spreadsheet control

You can use the `print` method by importing from `ej2-base` package. Here, the `Select` event in the dropdown and the `dataBound` event are used to print the single/multiple sheets of data. To print the single/multiple sheets, use the dropdown button and select the `Print` (or) `Print All` option. In the following sample, you can be able to print the single/multiple sheets.

INDEX.TS

```

import { createElement, print } from '@syncfusion/ej2-base';
import { Spreadsheet, SheetModel, MenuSelectEventArgs, UsedRangeModel } from
 '@syncfusion/ej2-spreadsheet';
import { budgetData, salaryData } from './datasource.ts';
import { DropDownButton, ItemModel, MenuEventArgs } from "@syncfusion/ej2-
splitbuttons";
//Initialize action items.
let items: ItemModel[] = [
    {
        text: "Print"
    },
    {

```

```

        text: "Print All"
    }
];
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton({
    items: items,
    cssClass: "e-round-corner",
    select: (args: MenuEventArgs) => {
        if (args.item.text === 'Print') {
            printElement.querySelector(".e-sheet-content").innerHTML =
document.querySelector(
                ".e-sheet-content"
            ).innerHTML; // To add the spreadsheet table
            let usedRange: UsedRangeModel =
spreadsheet.getActiveSheet().usedRange;
            let tbody: Element = printElement.querySelector('tbody');
            for (let i: number = tbody.getElementsByClassName('e-row').length; i
>= 0; i--) {
                if (tbody.getElementsByClassName('e-row')[i] &&
parseInt(tbody.getElementsByClassName('e-row')[i].getAttribute('aria-
rowindex')) > usedRange.rowIndex + 1) {
                    tbody.getElementsByClassName('e-row')[i].remove();
                }
            }
            (printElement.querySelector('.e-sheet-
content').children[0].getElementsByClassName('e-virtualtrack')[0] as
HTMLElement).style.height = 'auto';
            print(printElement);
            printElement.querySelector(".e-sheet-content").innerHTML = '';
        }
        if (args.item.text === 'Print All') {
            let sheets: SheetModel[] = spreadsheet.sheets;
            if (spreadsheet.activeSheetIndex === 0) {
                printElement.querySelector(".e-sheet-content").innerHTML =
document.querySelector(
                    ".e-sheet-content"
                ).innerHTML; // To add the spreadsheet table
                let usedRange: UsedRangeModel =
spreadsheet.getActiveSheet().usedRange;
                let tbody: Element = printElement.querySelector('tbody');
                for (let i: number = tbody.getElementsByClassName('e-row').length; i
>= 0; i--) {
                    if (tbody.getElementsByClassName('e-row')[i] &&
parseInt(tbody.getElementsByClassName('e-row')[i].getAttribute('aria-
rowindex')) > usedRange.rowIndex + 1) {
                        tbody.getElementsByClassName('e-row')[i].remove();
                    }
                }
                if (sheets[spreadsheet.activeSheetIndex + 1]) {
                    spreadsheet.goTo(sheets[spreadsheet.activeSheetIndex + 1].name +
"!A1");
                    isPrint = true;
                } else {
                    print(printElement);
                    printElement.querySelector(".e-sheet-content").innerHTML = '';
                }
            } else {

```



```

        if (sheets[0]) {
            spreadsheet.goTo(sheets[0].name + "!A1");
            isPrint = true;
        }
    }
}
});
// Render initialized DropDownButton.
drpDownBtn.appendTo("#element");
let printElement: HTMLElement = createElement("div", {
    className: "e-sheet-panel",
    innerHTML:
        '<div class="e-spreadsheet-print"></div><div class="e-sheet"><div
class="e-main-panel style="height:100%" style="overflow: unset"><div
class="e-sheet-content" ></div></div></div>'
}); // creating same hierarchy of element as DOM
let isPrint: boolean = false;
let columns: ColumnModel[] = [{ width: 100 }, { width: 100 }, { width: 100 },
    { width: 100 }];
let spreadsheet: Spreadsheet = new Spreadsheet({
    sheets: [{ name: 'Budget', ranges: [{ dataSource: budgetData }], columns:
columns },
        {name: 'Salary', ranges: [{ dataSource: salaryData }],
columns: columns}],
    created: (): void => {
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'Salary!A1:D1');
        spreadsheet.cellFormat({ fontWeight: 'bold'}, 'Salary!A7:D7');
    },
    dataBound: () => {
        if (isPrint) {
            printElement.querySelector(".e-sheet-content").innerHTML += document
                .querySelector(".e-sheet-content").outerHTML;
            let usedRange: UsedRangeModel =
spreadsheet.getActiveSheet().usedRange;
            let tbody: Element = printElement.querySelector('.e-sheet-
content').children[spreadsheet.activeSheetIndex].querySelector('tbody');
            for (let i: number = tbody.getElementsByClassName('e-row').length; i
>= 0; i--) {
                if (tbody.getElementsByClassName('e-row')[i] &&
parseInt(tbody.getElementsByClassName('e-row')[i].getAttribute('aria-
rowindex')) > usedRange.rowIndex + 1) {
                    tbody.getElementsByClassName('e-row')[i].remove();
                }
            }
            let sheets: SheetModel[] = spreadsheet.sheets;
            if (sheets.length - 1 === spreadsheet.activeSheetIndex) {
                let count: number = printElement.querySelector(".e-sheet-
content").childElementCount;
                if (count > 1) {
                    for (let i: number = 0; i < count; i++) {

```

```

        (printElement.querySelector('.e-sheet-
content').children[i].getElementsByClassName('e-virtualtrack')[0] as
HTMLElement).style.height = 'auto';
        printElement.querySelector('.e-sheet-
content').children[i].setAttribute('style', 'page-break-after: always;')
    }
}
print(printElement);
isPrint = false;
printElement.querySelector(".e-sheet-content").innerHTML = '';
} else {
    if (sheets[sheetIndex + 1]) {
        spreadsheet.goTo(sheets[sheetIndex + 1].name +
"!A1");
    }
}
}
}
});
//Render initialized Spreadsheet component
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

```

```

<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="dialog"></div>

  <div id="element">Print</div>
  <div id="container">
    <div id="spreadsheet"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Changing the active sheet while importing a file in EJ2 JavaScript Spreadsheet control

You can change the active sheet of imported file by updating [activeSheetIndex](#) property on the [openComplete](#) event.

The following code example shows how to set the active sheet when importing an Excel file.

INDEX.TS

```

import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let spreadsheet: Spreadsheet = new Spreadsheet({
  openUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/open',
  saveUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/save',
  openComplete: function() {
    if (spreadsheet) {
      spreadsheet.activeSheetIndex = 2;
    }
  }
});
spreadsheet.appendTo('#spreadsheet');

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 SpreadSheet</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="JavaScript UI Controls">
<meta name="author" content="Syncfusion">
<link rel="shortcut icon" href="resources/favicon.ico">
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<!--Element which is going to render-->
<div id="container">
<div id="spreadsheet"></div>
</div>
<script>
var ele = document.getElementById('container');
if (ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Import an excel document using file uploader in EJ2 JavaScript Spreadsheet control

If you explore your machine to select and upload an excel document using the file uploader, you will receive the uploaded document as a raw file in the [success](#) event of the file uploader. In this **success**

event, you should pass the received raw file as an argument to the Spreadsheet's [open](#) method to see the appropriate output.

INDEX.TS

```
import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { Uploader } from '@syncfusion/ej2-inputs';
let spreadsheet: Spreadsheet = new Spreadsheet({
    openUrl:
    'https://services.syncfusion.com/js/production/api/spreadsheet/open',
});
spreadsheet.appendTo('#spreadsheet');
let uploader: Uploader = new Uploader({
    asyncSettings: {
        saveUrl:
            'https://services.syncfusion.com/js/production/api/FileUploader/Save',
        removeUrl:
            'https://services.syncfusion.com/js/production/api/FileUploader/Remove',
    },
    success: onSuccess,
    allowedExtensions: '.csv, .xls, .xlsx',
});
uploader.appendTo('#uploader');
function onSuccess(args): void {
    if (args.operation == 'upload')
        spreadsheet.open({ file: args.file.rawFile });
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <div id="spreadsheet"></div>
    <input type="file" id="uploader" name="UploadFiles" />
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Identify the context menu opened in EJ2 JavaScript Spreadsheet control

The Spreadsheet includes several context menus that will open and display depending on the action. When you right-click on a cell, for example, a context menu with options related to the cell element appears.

The class name returned by the [contextMenuBeforeOpen](#) event can be used to identify the context menu that is opened. The context menus and their class names are tabulated below.

Class name	Context menu name
-----	-----
.e-sheet-content	Cell context menu
.e-toolbar-item	Footer context menu
.e-rowhdr-table	Row header context menu
.e-colhdr-table	Column header context menu

The following code example shows how to identify the context menu opened.

INDEX.TS

```

import { Spreadsheet } from '@syncfusion/ej2-spreadsheet';
import { enableRipple, closest } from '@syncfusion/ej2-base';

```

```
import { BeforeOpenCloseMenuEventArgs } from '@syncfusion/ej2-navigations';
enableRipple(true);
let spreadsheet: Spreadsheet = new Spreadsheet({
  contextMenuBeforeOpen: (args: BeforeOpenCloseMenuEventArgs): void => {
    if (closest(args.event.target, '.e-sheet-content')) {
      console.log('Cell Context Menu');
    } else if (closest(args.event.target, '.e-colhdr-table')) {
      console.log('Column Header Context Menu');
    } else if (closest(args.event.target, '.e-rowhdr-table')) {
      console.log('Row Header Context Menu');
    } else if (closest(args.event.target, '.e-toolbar-item')) {
      console.log('Footer Context Menu');
    }
  },
});
spreadsheet.appendTo('#spreadsheet');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>EJ2 SpreadSheet</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="JavaScript UI Controls">
  <meta name="author" content="Syncfusion">
  <link rel="shortcut icon" href="resources/favicon.ico">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>
```

Save and open Spreadsheet data as a Base64 string in EJ2 JavaScript Spreadsheet control

In the Spreadsheet control, there is currently no direct option to save and open data as a Base64 string. You can achieve this by saving the Spreadsheet data as blob data and then converting that saved blob data to a Base64 string using FileReader.

You can get the Spreadsheet data as blob in the [saveComplete](#) event when you set the `needBlobData` as `true` and `isFullPost` as `false` in the [beforeSave](#) event.

The following code example shows how to save and open the spreadsheet data as base64 string.

INDEX.TS

```
import { Spreadsheet, BeforeSaveEventArgs, SaveCompleteEventArgs } from
'@syncfusion/ej2-spreadsheet';
import { data } from './datasource.ts';
let base64String: string | ArrayBuffer;
let spreadsheet: Spreadsheet = new Spreadsheet({
  openUrl:
'https://services.syncfusion.com/js/production/api/spreadsheet/open',
  sheets: [{
    name: 'Car Sales Report',
    ranges: [{ dataSource: data }],
    columns: [
      { width: 180 }, { width: 130 }, { width: 130 }, { width: 180 },
      { width: 130 }, { width: 120 }
    ]
  }],
  created: (): void => {
    //Applies cell and number formatting to specified range of the active
    sheet
    spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center',
    verticalAlign: 'middle' }, 'A1:F1');
  },
  beforeSave: (args: BeforeSaveEventArgs): void => {
    args.needBlobData = true; // To trigger the saveComplete event.
    args.isFullPost = false; // Get the spreadsheet data as blob data in the
    saveComplete event.
  },
```



```

saveComplete: (args: SaveCompleteEventArgs): void => {
    // Convert blob data to base64 string.
    let reader: FileReader = new FileReader();
    reader.readAsDataURL(args.blobData);
    reader.onloadend = function () {
        base64String = reader.result ? reader.result : '';
    };
}
});
spreadsheet.appendTo('#spreadsheet');
document.getElementById("import")!.onclick = (): void => {
    // Open the file based on saved base64 string.
    fetch(base64String as string)
        .then((response) => response.blob())
        .then((fileBlob) => {
            let file = new File([fileBlob], 'Sample.xlsx');
            spreadsheet.open({ file: file });
        });
}
document.getElementById("export")!.onclick = (): void => {
    spreadsheet.save({
        url:
        'https://services.syncfusion.com/js/production/api/spreadsheet/save',
        fileName: 'Worksheet',
        saveType: 'Xlsx',
    }); // Specifies the save URL, file name, file type need to be saved.
    // Logs base64 string into the console.
    console.log('Base64 String - ', base64String);
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <button class="e-btn custom-btn" id="import">Import Base64</button>
    <button class="e-btn custom-btn" id="export">Export as Base64</button>
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Insert a sheet programmatically and make it the active sheet in EJ2 JavaScript Spreadsheet control

A sheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data. Using [insertSheet](#) method, you can insert one or more sheets at the desired index. Then, you can make the inserted sheet as active sheet by focusing the start cell of that sheet using the [goTo](#) method.

The following code example shows how to insert a sheet programmatically and make it the active sheet.

INDEX.TS

```

import { Spreadsheet, SheetModel, ColumnModel } from '@syncfusion/ej2-
spreadsheet';
import { data, employeeData } from './datasource.ts';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let columns: ColumnModel[] = [
  { width: 180 }, { width: 130 }, { width: 130 },
  { width: 180 }, { width: 130 }, { width: 120 }
];
let sheets: SheetModel[] = [{
  name: 'Car Sales Report',
  ranges: [{ dataSource: data }],

```

```

        columns: columns
    }];
    let spreadsheet: Spreadsheet = new Spreadsheet({
        sheets: sheets,
    });
    spreadsheet.appendTo('#spreadsheet');
    document.getElementById("insertSheet")!.onclick = () => {
        spreadsheet.insertSheet(
            [
                {
                    index: 1,
                    name: 'new_sheet',
                    ranges: [
                        {
                            dataSource: employeeData,
                            startCell: 'A1'
                        },
                    ],
                    columns: columns,
                },
            ],
        );
        // Use the timeout function to wait until the sheet is inserted.
        setTimeout(() => {
            // Method for switching to a new sheet.
            spreadsheet.goTo('new_sheet!A1');
        })
    };
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <button class="e-btn custom-btn" id="insertSheet">Insert Sheet</button>
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Get the filtered rows in EJ2 JavaScript Spreadsheet control

Filtering allows you to view specific rows in a spreadsheet while hiding the others. The [allowFiltering](#) property allows you to enable or disable filtering functionality through the UI. You can also use the [allowFiltering](#) property and [applyFilter](#) method combination to filter data via code behind. The filtered rows can be identified by iterating through the row collection on the sheet and using the `isFiltered` property available in each row object.

The following code example shows how to get the filtered rows.

INDEX.TS

```

import { Spreadsheet, ColumnModel, ExtendedRowModel, SheetModel,
UsedRangeModel } from '@syncfusion/ej2-spreadsheet';
import { PredicateModel } from '@syncfusion/ej2-grids';
import { defaultData } from './datasource.ts';
let columns: ColumnModel[] = [
  { width: 180 }, { width: 130 }, { width: 130 }, { width: 180 },
  { width: 130 }, { width: 120 }
]
let spreadsheet: Spreadsheet = new Spreadsheet({
  sheets: [{ ranges: [{ dataSource: defaultData }], columns: columns }],
  created: function () {
    // Applies cell formatting to specified range of the active sheet

```

```

        spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle' }, 'A1:F1');
        // Construct the predicate model to be updated to the data.
        let predicates: PredicateModel[] = [{
            field: 'C',
            operator: 'equal',
            value: 'Pink',
            matchCase: false
        }];
        // Apply filter to the specified range.
        spreadsheet.applyFilter(predicates, 'A1:C7');
    }
});
spreadsheet.appendTo('#spreadsheet');
document.getElementById("getFilterData").onclick = () => {
    let activeSheet: SheetModel = spreadsheet.getActiveSheet();
    let usedRange: UsedRangeModel = activeSheet.usedRange;
    for (let i: number = 0; i <= usedRange.rowIndex; i++) {
        // Get the filtered row using isFiltered property.
        let filteredRow: Object = (activeSheet.rows[i] as
ExtendedRowModel).isFiltered;
        if (!filteredRow) {
            let rowData: Object = spreadsheet.getRowData(i);
            console.log("Row:", i + 1, "Cells", rowData);
        }
    }
};

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>EJ2 SpreadSheet</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link rel="shortcut icon" href="resources/favicon.ico">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
spreadsheet/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.4.1/shim.min.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <!--Element which is going to render-->
  <div id="container">
    <button class="e-btn custom-btn" id="getFilterData">Get
Filtered Data</button>
    <div id="spreadsheet"></div>
  </div>
  <script>
    var ele = document.getElementById('container');
    if (ele) {
      ele.style.visibility = "visible";
    }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

Mobile responsiveness in EJ2 JavaScript Spreadsheet control

The Spreadsheet control rendered in desktop mode will be adaptive in all mobile devices where the layout gets adjusted based on their parent element's dimensions to accommodate any resolution.

You can see the overflowed items of ribbon header, ribbon content, and sheet tab using touch and swipe action. The right navigation arrow is added at the end of the ribbon content through which the user can navigate towards overflowed items. Once you reached the rightmost end of the ribbon content, the right navigation arrow will change to left navigation arrow through which you can navigate to the left of the ribbon content.



The screenshot shows an Excel spreadsheet with a ribbon at the top containing 'File', 'Home', 'Insert', 'Formulas', and 'Data'. The 'Formulas' tab is active, showing a formula bar with 'fx Customer Name'. Below the ribbon is a table with two columns: 'Customer Name' and 'Model'. The table contains 19 rows of data, numbered 1 to 19 in the first column. The last row is highlighted in pink. At the bottom of the table, there is a tab labeled 'Car Sales Report'.

	A	B
1	Customer Name	Model
2	Romona Heaslip	Taurus
3	Clare Batterton	Sparrow
4	Eamon Traise	Grand Cherokee
5	Julius Gorner	GTO
6	Jenna Schoolfield	LX
7	Marylynne Harring	Catera
8	Vilhelmina Leipelt	7 Series
9	Barby Heisler	Corvette
10	Karyn Boik	Regal
11	Jeanette Pamplin	S4
12	Cristi Espinos	TL
13	Issy Humm	Club Wagon
14	Tuesday Fautly	V8 Vantage
15	Rosemaria Thomann	Caravan
16	Lyell Fuentez	Bravada
17	Raynell Layne	Colorado
18	Raye Whines	4Runner
19	Virgina Aharoni	TSX

Stepper

Steps in EJ2 JavaScript Stepper control

The JavaScript Stepper allows you to add steps using the [steps](#) property. Each step can be configured with options such as `iconCss`, `text`, `label`, `cssClass` and more.

Adding steps

You can define the icon and text content for each step using the `iconCss`, `text` and `label` properties.

Defining icon CSS

You can define the CSS class to show the icon for each step using the `iconCss` property.

INDEX.JS

```
var iconOnly = [
  { iconCss: 'sf-icon-cart' },
  { iconCss: 'sf-icon-transport' },
  { iconCss: 'sf-icon-payment' },
  { iconCss: 'sf-icon-success' }
```

```

];
var stepperWithIcon = new ej.navigations.Stepper({
  steps: iconOnly
});
stepperWithIcon.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  margin-top: 30px;
  padding: 30px;
}
@font-face {
  font-family: 'Default';
  src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelIPaAAABmAAAAF5nbHlmEwr

```



```
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCoc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBgcFBQOEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgC/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQHCPCBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEAE8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHxYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHxYUEXEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsmDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFB1SEA8NCwoIBQQAQOFCaOLDQ8QEH1UFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBQADAgEBAQMEBAQGBQcGBwgICAgICAgIBwYHBQYEBQADAgH
+aAICAgHBAFYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAGQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDaf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgHCBwg
ICAKIBwgHBwYGBQUEAwMCAQECAGMEBQUGBgHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWQBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANKa5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDw0dAQ8BKwIvAT0BLwC9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQE
BAIkCAGIEAwQFDA0SBwcGAGIBAQICBgHbXyKCQkJCAcHBgUFBAQCAQECAGQIDAQWQBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECAGMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfXcWFxYXfHxYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwkRNRI
HBqADChI1DQoFAgEBAgMEBAOMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAQWUGBwgJCgICAQENAEFAwI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAGMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAQYDgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAQEBAGQICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBAQEBOQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQWUBgYICAKJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQwNgsMDgcIJCYNmyZOTYcmJiYlJSQjIiIgHwULCAMBAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAAAAAAABAAAAAABAAcAAQABAAAAAAACAACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQAABAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVKaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAAQBVAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYABwBuAHQAIAABnAGUAbgB1AHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAEQBuAGMAZgB1AHMAAQBVAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
```

```

AeQBAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAA
AAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydfF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Defining text content

You can define text instead of an icon by setting the `text` property and display label content for a step using the `label` property.

When both label and text are defined, the label takes priority for display based on the `stepType`.

INDEX.JS

```

var textOnly = [
    { text: 'A' },
    { text: 'B' },
    { text: 'C' },
    { text: 'D' }
];
var labelOnly = [
    { label: 'Cart' },
    { label: 'Delivery Address' },
    { label: 'Payment' },
    { label: 'Confirmation' }
];
var textStepper = new ej.navigations.Stepper({
    steps: textOnly,
    stepType: 'indicator'
});
textStepper.appendTo("#stepper");
var labelsStepper = new ej.navigations.Stepper({
    steps: labelOnly
});
labelsStepper.appendTo("#labelStepper");

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 - Stepper</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
        <nav id="labelStepper" style="margin-top: 50px;"></nav>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Optional steps

You can show whether the step is optional or not by using `optional` property. By default, the `optional` property is `false`.

INDEX.JS

```

var stepperIcon = [
    { iconCss: 'sf-icon-cart' },
    { iconCss: 'sf-icon-transport' },
    { iconCss: 'sf-icon-payment', optional: true },
    { iconCss: 'sf-icon-success' }
];
var optionalStepper = new ej.navigations.Stepper({
    steps: stepperIcon
});
optionalStepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  margin-top: 30px;
  padding: 30px;
}
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAMvMNTYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIuQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAABAAAAAQAAAAA
EAAAABAAAAAQAAAAEAAAAAAAAGAAAAAMAAAAUAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAA

```

```

AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQCAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDawLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CfHITE+VHDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEa5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAqQHCPCBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKcAYGBAL+KgEEBQgKcW0PEBETFBQWFxcXfHfYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERUMFBYWFxcXfHfQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCasMDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKcAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDByVFRQSERAPDQsKCAUEAQEEBQgKcW0
PEBESFBUVFhcXfXyVFBiSEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAWCAQECaWQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8CowEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BagoCagENDawKCgghBQE
BAikCagIEAwQFDA0SBwcGAgIBAQICBgcHBxYKQkKJCAcHBgUFBAMCAQEBAQIDAQWFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYQFBAQEBAgYHBwcWCgkKCAgYHwYFBQQAQAgE
BAQECAwMEBAyFBgcHEBABAQED/qwUFRUVFRYWFHfYWFxYXfHcXfXcWFxYXfHfYWFHfYFRUVFAQCAQI
CBAUGCagJCgsLDA0MDQ0NDBk2EQYgGqYGCESZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwKRNRI
HBqADChI1DQoFagEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAQWUGBwgJCgICAQENAEQFAwI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAyDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQIaAgEBAf6RDAsLCQkICAYGBQUdAwIBAQIDAQWFBgYICakJCwsMKsckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYNmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAACAAQABAAAAAACAACACAABAAAAAADAACADwABAAAAAEEAACAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZwZhdWx0UmVndWxhckR
lZmFlbHREZwZhdWx0VmVyc2lvbiAxLjBEZwZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctYU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQBByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBUAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBwAGMAZgBlAHMAaQBvAG4ALgBjAG8ABQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAA
AAAAAAYBAGEDAQQBBQEGAQCADXRYW5zcG9ydC12YW4ldXNlciltb2RpZnRlc2hvchBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
}

```

```

speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Disabling steps

You can use the `disabled` property to disable a step, preventing user interaction when set to `true`. By default, the value is `false`.

INDEX.JS

```

var stepperIcon = [
  { iconCss: 'sf-icon-cart' },
  { iconCss: 'sf-icon-transport' },
  { iconCss: 'sf-icon-payment', disabled: true },
  { iconCss: 'sf-icon-success' }
];
var stepper = new ej.navigations.Stepper({
  steps: stepperIcon
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAGAAAAAAMAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAGIBgcFBQOEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLcwsJCQkHBwUFawKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLcgoJCAGGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBGgH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQqHCPcBAGQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEAs8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEAs8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBiSEA8NCwoIBQQAQqFCAoLDQ8QEHlUFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgH

```

```
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECawQEBAYFBwYHCAGICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECawMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQFQBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAGENDAwKCggHBQE
BAikCAGIEAwQFDA0SBwcGAGIBAQICBgCHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAQFQBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECawMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaiIDAUGBwgJCgICAQENAEFAwI
DAgEAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAGICQgHBwcGBgYFBQQEBAQIDAQFQBQYGBgc
GBgUFBAQEBAQIAGEBaf6RDAsLCQkICAYGBQUDAwIBAQIDAQFQBQYICAKJCwsMKsckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwWNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwCYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYS
EJwwGAGIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiwADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQBLAGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBIAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgBIAHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgBIAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaAaAaAaAaAaAaAaAaA
AAAAAAAYBAGEDAQBBQEGAQcADXRYYW5zcG9ydC12YW4LdXNlcil1b2RpbmctY2FyY2F8wMS0Lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }
```

Setting active step

You can set the active step by specifying its index using the [activeStep](#) property. The default value is 0.

INDEX.JS

```
var iconOnly = [
{ iconCss: 'sf-icon-cart' },
```



```

    { iconCss: 'sf-icon-transport' },
    { iconCss: 'sf-icon-payment' },
    { iconCss: 'sf-icon-success' }
  ];
  var stepperWithIcon = new ej.navigations.Stepper({
    steps: iconOnly,
    activeStep: 1
  });
  stepperWithIcon.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  margin-top: 30px;
  padding: 30px;
}
@font-face {

```

```
font-family: 'Default';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKIAAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIIABOCc5wbnCOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBGcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQcHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICACGBGQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwC
GBGUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICACGBQQAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBGcH/iwNDawLCwoJCQgHBGUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgyEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAGFBgIWAQgHCPcBAGQGBGgKCgsMDQ4PDxQEBApDw4NDAsLCQgGBGQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWfXcXfHwYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWfXcXfHwYUExEQDw0LCgGfBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQDQAgECAGMEBAUGBgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBGZ
qBgOHBQMDMAMHBWMAWQICBQYKChYCB1wICBAPDw4NDAsLCQgGBGQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcxYfYVFBISEA8NCwoIBQQBAQFCAoLDQ8QEhIUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBGQEBAQCAQECAGwQEBAYFBwYHCAgICAgICAcIBGcFBGQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBGUFBAQCAgEBAQECAGQEBQUGBgHwG
ICAkIBwGHBwYGBQUEAwMCAQECAGwMEBQUGBGcHCAcICQgICACHBwYGBQUEBAICAEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBGyYFBQQDAwIBAQIDAQwQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQE
BAIkCAGIEAwQFDA0SBwcGAGIBAQICBgHbXyKcQkJCAcHBGUFBAQCAQECBAQIDAQwQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECAGwMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfXcWFxYXfHwYFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBGUEAgIBaQQAQECBAwKRNRI
HBqADChI1DQoFAGEBAGMEBAOMew8eTw4IVxkXCwkJBwYCOAIBAIIDAUGBwGJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAGwMEBAUGBgYHCAgICQgHBwCGBGyYFBQQEBAQYDIgICAQECaI
CBAUGBwGJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAQEAQICBAMFBQUGBwCICAgJBwGHBGc
GBGUFBAQEQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQwUFBgYICAKJCwsMKSckIiAeGxOYfHq
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwNdxAREXQWGBobHiAiJcCoAMDawQEA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwYAwEaBwQBAQIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDGcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAAAAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAAAADAACADwABAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAABAAAAAABAAcAwLwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAYwADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcmcGU3luY2Z
```

```

1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbnW9uZXkFY2hlY2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Step status

Each step's progress state can be specified using the `status` property. The possible values are `NotStarted`, `InProgress` and `Completed`. By default, the value is `NotStarted`.

INDEX.JS

```

var iconWithLabel = [
    { label: 'Cart', iconCss: 'sf-icon-cart' },
    { label: 'Payment', iconCss: 'sf-icon-payment' },
    { label: 'Ordered', iconCss: 'sf-icon-success' }
];
var stepper = new ej.navigations.Stepper({
    steps: iconWithLabel,
    stepChanged: () => {
        var status = stepper.steps[1].status
        updateStatus(status)
    }
});
stepper.appendTo('#stepperStatus');
function updateStatus(stepStatus) {
    var statusMap = {
        'NotStarted': { text: 'Your payment has not started yet', color:
'#e74d4d' },
        'InProgress': { text: 'Processing your payment', color: 'orange' },
        'Completed': { text: 'Payment successful', color: '#4CAF50' }
    };
    var currentStatus = document.getElementById("paymentStatus");
    if (currentStatus) {
        var { text, color } = statusMap[stepStatus];
        currentStatus.innerText = text;
    }
}

```

```
        currentStatus.style.backgroundColor = color;
    }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id="container">
    <div class="stepper-section">
      <nav id="stepperStatus"></nav>
      <div id="paymentStatus">Your payment has not started yet</div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#paymentStatus {
  padding: 10px;
  background-color: #e74d4d;
  color: white;
  border-radius: 5px;
  margin: 10px auto;
  text-align: center;
  font-weight: bold;
  max-width: 350px;
```

```
margin-top: 50px;
}
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1lUf5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAagAAAAAAAUAAAAQAABQABABKAAAADAAIAAIABOCc5wbnCOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAg8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKyp/10ICAcGBQQAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEFAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgY
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQOHCPCBAQOGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GBAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCA5MDQ4
IERESFBQUFQDAGECAGMEBAUGBgYIBwgJCQoKCwsLDawMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMMDAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEHlUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAGEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAGH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAGQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDaf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAGMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANKa5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY8HDWYdAR8WDw0daQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQE
BAIkCAGIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCACHBgUFBAMCAQEBQIDAwQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIBwYCAQEBQEBAGYHBwCWCgkKCAgHBwYFBQQDAGe
BAQECAGMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXFhCXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAgQCAQEBAwkRNRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAIIDAwUGBwgJCgICAgENAQEFawI
DAGECAGMFAwMEBAUDAMFAwIBAQECAGMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAQYDIgICAgECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAGeAwQEAQICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBAQEBQQAiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICAKJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwcYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAAI
```

```

AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAbnAGUAbgBlAHIAZQB0AGUAZAAgAHUAacwBpAG4AZwA
gAFMAeQBvAG4AIAbnAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAG4AIAbnAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkRc2hvcHBpYmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('true');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }
.stepper-section {
width: 75%;
margin: 0px auto;
min-width: 85px;
padding: 25px 0;
}

```

Step styling

You can use the `cssClass` property to customize the appearance of each step.

INDEX.JS

```

var stepperIconLabel = [
{ iconCss: 'sf-icon-cart', label: 'Cart' },
{ iconCss: 'sf-icon-transport', label: 'Delivery Address' },
{ iconCss: 'sf-icon-payment', label: 'Payment', cssClass: 'custom-step',
optional: true},
{ iconCss: 'sf-icon-success', label: 'Confirmation' }
];
var stepper = new ej.navigations.Stepper({
steps: stepperIconLabel
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 - Stepper</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAaZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAAMAAAAUAMAAQAAABQABABKAAAADAAIAAIABoCc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxAqDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk

```

```

IBwcHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQCAQ0NDQwLCgoJCAgGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCH/iwNDawLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLWejDwE1LwMjNz0BJZcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQOHCPCBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDawMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECaWQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICAKIBwgHBwYGBQUEAwMCAQECaWMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDAwQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDw0daQ8BKwIvAT0BLwC9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQE
BAikCAGIEAwQFDA0SBwCGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAQCAQEBQIDAQWQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECaWMEBAyFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfXcWFxYXfHYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAQICAQEBAwKRNRI
HBqADChI1DQoFAgEBAGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAUGBwgJCgICAQENAQEFaWI
DagECAGMFAwMEBAUDBAMFAwIBAQECAWMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAYDIgICAQECaIi
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAQEBAGICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBAQEBOQiAgEBAf6RDAsLCQkICAYGBQUdAwIBAQIDAwUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQOnEAKKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctYU3luY2Z
1c2lvbiBNZXRybyBTdHVKaW93d3cuc3luY2Z1c2lvbi5jb2AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgB1AHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAOAAAAA
AAAAAAAAAYBAGEDAQBBQEGAQcADXRYYW5zcG9ydC12YW4LdXNlcil1tb2RpZnRc2hvcHBpbmctY2F
ydf8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueType');

```

```

font-weight: normal;
font-style: normal;

```

```

}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;

```



```

    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }
#stepper .custom-step .e-step-label-optional {
    font-style: italic;
    font-weight: 900;
    color: #299100;
}

```

Step validation

You can set the validation state for each step to displaying a success or error icon by using `isValid` property.

To know more about Stepper validation, please refer to the [Validation](#) section.

Step types in EJ2 JavaScript Stepper control

The Stepper control provides support for displaying steps with the following step types.

Default type

In default type, the Stepper displays steps with a combination of both indicators and labels by setting the `stepType` property as `Default`. By default, the Stepper displays steps in the `Default` type.

INDEX.JS

```

var iconWithLabel = [
    { label: 'Cart', iconCss: 'sf-icon-cart' },
    { label: 'Delivery Address', iconCss: 'sf-icon-transport' },
    { label: 'Payment', iconCss: 'sf-icon-payment' },
    { label: 'Confirmation', iconCss: 'sf-icon-success' }
];
var stepper = new ej.navigations.Stepper({
    steps: iconWithLabel,
    stepType: 'default'
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    text-align: center;
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EAAAABAAAAQAQAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkKJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw

```

```

D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCkAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCkAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgC/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAqQHCPCBAgQGBgKCGsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEAS8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCgqFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDawMDQ0ODQ4PDgEzawIBAQIGBQMCAQDBgZ
qBgOHBQMDMAMHBwMMAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEAS8PDg0MCwoKCAyGawMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgCfBgQEBAMCAQECawQEBAYFBwYHCAGICAgICAcIBgCfBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECawMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQFQBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAGENDAwKCggHBQE
BAikCAGIEAwQFDA0SBwcGAGIBAQICBgCHBxYKQKJCAcHBgUFBAMCAQEBAQIDAQFQBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECawMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXFhCWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAQUGBwgJCgICAQENAEFAwI
DAgECAGMFAwMEBAUDBAwMFAwIBAQECawMEBAUGBgYHCAGICQgHBwCGBgYFBQQUEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAgEBAQICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBQEBQqiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQUGBwYICAkJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwYAwEaBwQBAgIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAGIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgSMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAACACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAYwADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctYU3luY2Z
lc2lvbiBNZXRybyBTdHVKaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQBLAGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBIAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgBIAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgBIAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
AAAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1b2RpbmctY2FyY2FwMS0Lc3BlbmQtbW9uZkFyY2h1Y2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;

```

```

text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Label type

In label type, the Stepper displays the steps with only the step labels by setting the [stepType](#) property as **Label**.

When both label and text are defined, the label takes priority in displaying the steps.

INDEX.JS

```

var iconWithLabel = [
  { label: 'Cart', iconCss: 'sf-icon-cart' },
  { label: 'Delivery Address', iconCss: 'sf-icon-transport' },
  { label: 'Payment', iconCss: 'sf-icon-payment' },
  { label: 'Confirmation', iconCss: 'sf-icon-success' }
];
var stepper = new ej.navigations.Stepper({
  steps: iconWithLabel,
  stepType: 'label'
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    text-align: center;
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXcDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUblAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAAUAUAMAAQAAABQABABKAAAADAAIAAIIABOcC5wbnCoc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAAAAEABAACAAMABQAAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAk
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABBgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBawMDBGIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAghQCPcBAGQGBgkKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhCFBAMDrxYWDQEBawUHCasMDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ

```

```

qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAWMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFB1SEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBaGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHbGcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCACHBgYFBQQDAwIBAQIDAQFQBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BagoCagENDAwKCcgHBQE
BAIkCAgIEAwQFDA0SBwCgAgIBAQICBgcHBxYKCQkJCACHBgUFBAMCAQEBAQIDAQFQBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHhYWFhYVFRUVFAQCAQI
CBAUGCagJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwkrNRI
HBqADChI1DQoFagEBaGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAuUGBwgJCgICAQENAQEFawI
DAgECagMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBagEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAQEBQICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBAQEBOQIAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQFQBQYICAKJCwsMKsckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYS
EJwwGAgIwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNGsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAACAAQABAAAAAACAACACAABAAAAAADAACADwABAAAAAEEAACAFgA
BAAAAAFAAAsAHQABAAAAAAGAACAkAABAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctU3luY2Z
lc2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgB1AHIAAYQB0AGUAZAAGAHUAACwBpAG4AZwA
gAFMAEQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuaGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaA
AAAAAYBAGEDAQBBQEGAQCADXRYW5zcG9ydC12YW4ldXNlcil1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Label positions

You can display the label on the top, bottom, left, or right side of the steps using the [labelPosition](#) property.

The following label positions are supported in Stepper:

Value	Description
Top	Positions the label at the top of each step.
Bottom	Positions the label at the bottom of each step.
Start	Positions the label to the left side of each step.
End	Positions the label to the right side of each step.

INDEX.JS

```
var iconWithLabel = [
  { label: 'Cart', iconCss: 'sf-icon-cart' },
  { label: 'Delivery Address', iconCss: 'sf-icon-transport' },
  { label: 'Payment', iconCss: 'sf-icon-payment' },
  { label: 'Confirmation', iconCss: 'sf-icon-success' }
];
var stepper = new ej.navigations.Stepper({
  steps: iconWithLabel
});
stepper.appendTo("#stepper");
window.updateLabelPosition = function(args) {
  stepper.labelPosition = args.value;
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="e-btn-group">
            <input type="radio" id="start" name="position" value="start"
onclick="updateLabelPosition(this);" />
            <label class="e-btn" for="start">Start</label>
            <input type="radio" id="end" name="position" value="end"
onclick="updateLabelPosition(this);" />
            <label class="e-btn" for="end">End</label>
            <input type="radio" id="top" name="position" value="top"
onclick="updateLabelPosition(this);" />
            <label class="e-btn" for="top">Top</label>
            <input type="radio" id="bottom" name="position" value="bottom"
onclick="updateLabelPosition(this);" checked/>
            <label class="e-btn" for="bottom">Bottom</label>
        </div>
        <div class="stepper-section">
            <nav id="stepper"></nav>
        </div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    text-align: center;
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKIAAAwAgTlMvMjlvSgcAAAEoAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYw1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAABwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAaZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxdDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8

```



```

NAR8FFSM1JxEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAk
JCgoKCQoICQgHBwCQBQDawEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ40DQ4MDAwLCgkJCAYGBQMDKAgiBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgcH/iwNDawLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg40Dg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBAMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQqHCPcBAgQGBgkKCsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhCFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQQAQAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ00DQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA00Dw8QATMLDByVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQAQQFCAoLDQ8QEHlUFYRYAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0raAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDagEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDagH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECaWQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICaSH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCGBwg
ICAkIBwgHBwYGBQUEAwMCAQECaWMEBQUGBgCHCAcICQgICACHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQAQwIBAQIDAQwQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY8HDwYdAR8WDw0dAQ8BKwIvAT0BLwC9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BagoCAGENDAwKCgghBQE
BAikCAGIEAwQFDA0SBwCgAgIBAQICBgcHBxYKQKJCACHBgUFBAMCAQEBAQIDAQwQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQAQAgE
BAQECaWMEBAQYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfXcWFxYXfHYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYgqYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAQQAQEBAwKRNRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAQUGBwgJcGICAQENAEQFAwI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAWMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAQYDgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAwQEAQEBAGICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBAQEBOQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQwUFBgYICakJCwsMKsckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNGsMDgcIJCYnmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAAAADAACADwABAAAAAAAEAAAFgA
BAAAAAAFAAAsAQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZwZhdWx0UmVndWxhckR
lZmFlbHREZwZhdWx0VmVyc2lvbiAxLjBEZwZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctYU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgB1AHIAAYQB0AGUAZAAGAHUAacwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABnAGUAdABYAGAAIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8ABQAAAAAACAAGAAAAAaAaAaAaAaAaAaAaAaAaAaAaA
AAAAAAYBAGEDAQBBQEGAQcADXRYyW5zcG9ydC12YW4LdXNlci1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;

```

```

font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }
.stepper-section {
    margin: 50px 100px;
}

```

Indicator type

In indicator type, the Stepper displays steps with only the step indicators by setting the [stepType](#) property as **Indicator**.

INDEX.JS

```

var textOnly = [
    { text: '1' },
    { text: '2' },
    { text: '3' },
    { text: '4' }
];
var defaultStepper = new ej.navigations.Stepper({
    steps: [{}, {}, {}, {}],
    stepType: 'indicator'
});
defaultStepper.appendTo("#stepperDefault");
var textStepper = new ej.navigations.Stepper({
    steps: textOnly,
    stepType: 'indicator'
});
textStepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->

```

```

<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepperDefault"></nav>
        <nav id="stepper" style="margin-top: 50px;"></nav>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Orientations in EJ2 JavaScript Stepper control

The Stepper control supports the display of steps in both horizontal and vertical orientations by using the [orientation](#) property.

Horizontal

In horizontal orientation, the steps are displayed in a side-by-side manner by setting the [orientation](#) property to **Horizontal**. By default, the steps are displayed in the horizontal orientation.

INDEX.JS

```

var iconOnly = [
    { iconCss: 'sf-icon-cart' },
    { iconCss: 'sf-icon-transport' },
    { iconCss: 'sf-icon-payment' },
    { iconCss: 'sf-icon-success' }
];
var horizontalStepper = new ej.navigations.Stepper({
    steps: iconOnly,
    orientation: 'horizontal'
});
horizontalStepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - Stepper</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAABwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAAMAAAUAAQAABQABABKAAAADAAIAAIABoCc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAAMABQAAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkKJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ40DQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc

```

```

GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICaCGBQQCAQ0NDQwLcGoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgcH/iwNDawLCwoJCQgHBgUEAwEBAQEEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BjZcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCwHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQgHCPcBAgQGBGgKCgSMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQEcbAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCasMDQ4
IERESFBQUFQODAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQODBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQEcbAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKCAYGawMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAaaaaa/QRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBGcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwg
ICakIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQODAwIBAQIDAwQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgCHBxYKQKJCAcHBgUFBAQCAgEBAQIDAwQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQODAgE
BAQECAwMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXFxcWFxYXfHwYFhYVFRUVFAQCAQI
CBAUGCAgJCgSLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRI
HBqADChI1DQoFagEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAUGBwgJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAyDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAQEBQICBAMFBQUGBwCICAgJBwgHBGc
GBgUFBAQEBOQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQYFBQqNEakKCwNDYxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNGsMDGcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctY3luY2Z
1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgB1AHIAAYQB0AGUAZAAgAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALGbjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAA
AAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4ldXNlcil1b2RpbmctY2FyY2F8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;

```

```

        -moz-osx-font-smoothing: grayscale;
    }
    .sf-icon-cart:before { content: "\e710"; }
    .sf-icon-transport:before { content: "\e702"; }
    .sf-icon-payment:before { content: "\e706"; }
    .sf-icon-success:before { content: "\e715"; }

```

Vertical

You can display the steps one below the other vertically by setting the [orientation](#) property to **Vertical**.

INDEX.JS

```

var iconOnly = [
    { iconCss: 'sf-icon-cart' },
    { iconCss: 'sf-icon-transport' },
    { iconCss: 'sf-icon-payment' },
    { iconCss: 'sf-icon-success' }
];
var verticalStepper = new ej.navigations.Stepper({
    steps: iconOnly,
    orientation: 'vertical'
});
verticalStepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 - Stepper</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
    <meta name="description" content="Essential JS 2">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
    <!--style reference from app-->
    <link href="styles.css" rel="stylesheet">
    <!--system js reference and configuration-->

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
    </div>
</body>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
margin-top: 30px;
padding: 30px;
min-height: 450px;
display: flex;
justify-content: center;
}
@font-face {
font-family: 'Default';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAEEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAaGAAAAAUAUAAAAQAABQABABKAAAADAAIAAIABoCc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAawADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAiwc3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAGIBGcFBQQEAgH+CwEBAGQEBQUHBggICAk
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQcHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ40DQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkKJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQCAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAGJCgoLDA0NDQCEBAUGBwQI1fo
BAgQFBGcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAaAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEFAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg40Dg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQHCPCBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQE8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQDAGECAGMEBAUGBgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QE8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAGEBAGMEBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAGH
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBGQEBAMCAQECawQEBAYFBwYHCAGICAgICAcIBGcFBGQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBGUFBAQCAgEBAQECAGQEBQUGBGcHBwG
ICAkIBwGHBwYGBQUEAwMCAQECawMEBQUGBGcHCAcICQgICAcHBwYGBQUEBAICAQEBQICBAQFBQY
GBwCHCAgICQgHCAcHBGyYFBQQDAwIBAQIDAQWFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAA

```

```

AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BagoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgCHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAQFQBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECaWMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFXcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwKRNRI
HBqADChI1DQoFAgEBaGMEBAoMEw8eTw4IVxkXCwkJBWYCOAIBaIIDAuUGBwgJCGICAQENAQEFaWI
DAgECAGMFAWMEBAUDBAMFAWIBAQECaWMEBAUGBgYHCAGICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJQCMBAgEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBQEBAQQAiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAuUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQOnEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwAlwABAAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lubiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYABwBuAHQAIAABnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaAaAaAaAaAaAaAaAaAa
AAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4ldXNlcil1b2RpbmctY2FyF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Linear flow in EJ2 JavaScript Stepper control

The Stepper control enables users to progress sequentially through each step, ensuring navigation from one step to the next in a linear way by setting the [linear](#) property to **true**. The default value is **false** allowing navigation between any steps and vice versa.

The example demonstrates the functionality of both linear and non-linear flow in the Stepper.

INDEX.JS

```

var iconWithLabel = [
{ label: 'Cart', iconCss: 'sf-icon-cart' },

```



```

{ label: 'Delivery Address', iconCss: 'sf-icon-transport' },
{ label: 'Payment', iconCss: 'sf-icon-payment'},
{ label: 'Confirmation', iconCss: 'sf-icon-success' }
];
var linearStepper = new ej.navigations.Stepper({
  steps: iconWithLabel,
  linear: true
});
linearStepper.appendTo("#stepper");
window.updateLinear = function(args) {
  linearStepper.linear = (/true/).test(args.value) ? true : false;
  linearStepper.reset();
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-btn-group">
      <input type="radio" id="linear" name="Linear" value="true"
onclick="updateLinear(this)" checked/>
      <label class="e-btn for="linear">Linear</label>
      <input type="radio" id="nonLinear" name="Linear" value="false"
onclick="updateLinear(this)" />
      <label class="e-btn for="nonLinear">Non-Linear</label>
    </div>
    <div class="stepper-section">

```

```

        <nav id="stepper"></nav>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
text-align: center;
margin-top: 30px;
padding: 30px;
}
@font-face {
font-family: 'Default';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAA
EAAAABAAAAQAQAAAEAAAAAAGAAAAAUAAMAAQAABQABABKAAAADAAIAAIABoCc5wbnCoc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAiwc3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAGIBgcFBQOEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICACGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAGIBwYGBAUDAQICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICACGBQQAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcFawUhLwIHly8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIwAgQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHwYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHwYUExEQDw0LCgGFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQDAGECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMMAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEHwYUFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAGM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAGEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAGH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECawQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAGQEBQUGBgcHBwg

```

```

ICAkIBwgHBwYGBwQWUEAwMCAQECAwMEBQUGBqgHCACICQgICACHBwYGBQUEBAICAQEBAQICBAQFQBQY
GBwcHCAgICQgHCACHBgYFBQQDAwIBAQIDAQWQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAAADAAA
AAANKa5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDW0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgqHBQE
BAikCAgIEAwQFDA0SBwCgAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAQWQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBHwYCAQEBAQEBAQYHbWcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXfhcXFxcWFxYXfYhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAQQCAQEBAwKRNRI
HBqADChILDQoFAgEBAgMEBAoMEW8eTw4IVxkXCwkJBWYCOAIBAiIDAuUGBwgJCgICAQENAEFAwI
DAgECAgMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAGICQgHBwCGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBAQEBOQIAGeBAf6RDAsLCQkICAYGBQUDAwIBAQIDAuUFBgYICAKJCwsMKsCKIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBBQnEAkKCwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDAwMCAQICAwCYawEaBwQBAGIAAAEAAAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIiJkIJAPrwB8AQmCQM
BARQuNGsMDgcIJCYnmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAAABAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAGAacAKAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhcR
lZmFlbHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctY3luY2Z
lc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQBIAgWAdABEAGUAZgBhAHUAbAB0AFYAZQBIAHMAaQBVAG4AIAAXAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAIBnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBUAGMAZgBlAHMAaQBvAG4AIAIBNAGUAdABYAG8AIAIBTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBUAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaA
AAAAAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1b2Rpb2RpbmctY2F
ydF8wMS0lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }
.stepper-section {
margin: 50px 100px;
}

```

Steps validation in EJ2 JavaScript Stepper control

The Stepper control allows you to set the validation state for each step, displaying either a success or error icon. You can define the success state of a step by setting the `isValid` property to `true`. If set to `false`, the step will display an error state. By default, the `isValid` property is `null`.

Based on the **stepType**, the validation state icon will be displayed either as an indicator or as part of the step label/text.

INDEX.JS

```
var iconOnly = [
  { iconCss: 'sf-icon-cart', isValid: true },
  { iconCss: 'sf-icon-transport' },
  { iconCss: 'sf-icon-payment', isValid: false },
  { iconCss: 'sf-icon-success' }
];

var iconStepper = new ej.navigations.Stepper({
  steps: iconOnly
});
iconStepper.appendTo("#stepper");
var labelOnly = [
  { label: 'Cart', isValid: true },
  { label: 'Address' },
  { label: 'Payment', isValid: false },
  { label: 'Confirmation' }
];

var labelsStepper = new ej.navigations.Stepper({
  steps: labelOnly
});
labelsStepper.appendTo("#labelStepper");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
```

```

        <nav id="labelStepper" style="margin-top: 50px;"></nav>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAMAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBGcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQcHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ40DQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQCAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAGJCgoLDA0NDQCEBAUGBwQI1fo
BAGQFBGcH/iwNDawLCwoJCQgHBgUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQHCPCBAgQGBggKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQCECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKcAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGAkMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQDAGECAGMEBAUGBgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQCECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKcAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAGEBAGMEBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAGH
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBgQEBAMCAQECAwQEBAYFBwYHCAGICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBGcHBwG
ICAkIBwGHBwYGBQUEAwMCAQECAwMEBQUGBGcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAA

```

```

AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BagoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAQWQBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECaWMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwKRNRi
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBWYCOAIBAiIDAuUGBwgJCgICAQENAQEFaWI
DAgECAGMFAWMEBAUDBAMFAWIBAQECaWMEBAUGBgYHCAGICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBQEBAQQAiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAuUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIcCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAACAAQABAAAAAACAACACAABAAAAAADAACADwABAAAAAEEAACAFgA
BAAAAAFAAAsAHQABAAAAAAGAACAkAABAAAAAAKACwAlwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lubiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYABwBuAHQAIAABnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
AAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4ldXNlcil1b2RpbmctY2FyF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Template in EJ2 JavaScript Stepper control

The Stepper control allows you to customize the default appearance and content of each step, creating a personalized experience for the user. You can use the [template](#) property to set the template content for the steps.

The step model and current step index are passed as `step` and `currentStep` properties in the template context for customization.

INDEX.JS

```
var iconWithLabel = [
```

```

    { label: 'PowerPoint', iconCss: 'sf-icon-powerpoint' },
    { label: 'Presentation', iconCss: 'sf-icon-projector' },
    { label: 'Backup', iconCss: 'sf-icon-onedrive' }
  ];
  var stepper = new ej.navigations.Stepper({
    activeStep: 1,
    steps: iconWithLabel,
    template: '<div class="template-content"><span
class="${step.iconCss}"></span><br><span class="e-
label">${step.label}</span></div>'
  });
  stepper.appendTo('#stepperTemplate');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="default-stepper-section">
      <nav id="stepperTemplate"></nav>
    </div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.template-content {
    background: #fff;
    width: 65px;
}
/* Stepper progressbar customization */
.e-stepper .e-stepper-progressbar {
    height: 3px;
    top: 25px;
}
.e-stepper .e-stepper-progressbar .e-progressbar-value {
    background-color: #27d96d;
}
/* Stepper status customization */
#stepperTemplate .e-step-completed * {
    color: #19cd60;
}
#stepperTemplate .e-step-inprogress * {
    color: #3479f3;
}
#stepperTemplate .e-step-notstarted * {
    color: #bdbdbd;
}
@font-face {
    font-family: 'template';
    src:
    url(data:application/x-font-ttf;charset=utf-
    8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfUAAAEoAAAAVmNtYXNlE+dkAAABlAAADxnbHlm39z
    zMQAAAdwAAAacaGVhZCaImHMAAADQAAAAANmhoZWEIUQQGAAARAAAACRobXR4FAAAAAAAYAAAA
    UbG9jYQR8BVAAAAHQAAAADG1heHABFwEbAAABCAAAACBuYW1ldkXdggAACHgAAAKRcG9zdD2fuhI
    AAAsMAAAAXwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABQABAAAAQAAPDfGf18
    PPPUACwQAAAAAOG7KcEAAAAA4bspwQAAAAAD9AP0AAAAACAACAAAAAABQAAAAAFAQ8ACAAAAA
    AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
    AAAAAAUGZFZABA5wDnAwQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
    EAAAAAABAAAAAQAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
    AAEABAAAAEAAgADAAQAAAAAAVQCagMoA04ACAAAAAAD0wP0AB8APABcAHwA+gD+AQoBDgAAAQ8
    HLWY9AT8GHwY1DwcvBj0BPwI7AR8FNw8HLWY9AT8GHwY1DwYrAS8GPwczHwUHFR8HMzcXDwIfBz8
    HJzcfAz8CFw8BHwc/By8HDwMnPW11LwcPBxUfAQcvBCMPaIc/Ay8HDwY1ESERAYEHFzczFzcnIRE
    hJyE1IQMiAQECAwQEBQUBAUDAwICAgIDAUEBQUBAQDAGH+mwEBAGMEBAUFBQQFAwMCAgILBgU
    FBQQEAwIBnAEBAGMEBAUFBQQEBAMCAgICAwQEBAGMEBAUFBQQEAwIB/rkBAQMDAwUEBQUBAQDAG
    EBAQECAwQEBQUBAUDAwMBUAEDBgYJCQsLCAXoAgQBAQQFBwgJCwsLCwkJBgYDAQFECAGICAKIBIEBAQE
    DBQCICGoLDAoJCQCFAwEBawUHCQkKDAoJCQ14BQMCAQMFBwgKCgwLCGoIBwUDAQICPwEJCQoMBwc
    GCGQFAwIBAQQFBwgJCwsLCwkJBgUEAsb89j8BZ8sr+gX7K8sBZ/x4DwOm/FoB7QUFBAQDAG
    EBAQECAwQEBQUBAUDAwMBAQEBAwMDBQQzBQUEBAMCAQEBAQIDBAQFBQUGCwICAgMEBARgBQUEAwMDAQE
    BAQMDAwQFBQUBAMDawEBAQEDAwMEBUSFBAQEAWICAgIDBAQEBAQDAGEBAGIDBAQFBQUGCgo
    IBwUDAQNNBASLCwsJCQYFBAEBawYGCQkLCwkuBAMCAQECAVkfCwsLCQgHBQQAQFBwgJCwsLCwk
    JBgYDAQECBQZTCACJCAsLCQgHBQQAQFBwgJCwsJCAUqAgcFAwEBAGrkCQgICAwKCggHBQMBQOM
    FBwgKCMp97gIS/bDALE3tLcACji8+AAAGAAAAAP0A+QAEwA5AEsAdwCFAIkAAAEzPwclLwcjFxU
    PDisBFsMRNx8OBsm1Mx8Njz8CFTMVDw4vAxUzFSMVMxUjFSE/AhEvAiE1IREzPwMRLwMhJREFEQE
    QIgcHBgUFBAECAQIEBQUGBAYmiAECAGQEBQYGCACJCAKkCQokNlSMCwoKCQgHBwYFBAMDAGEBgn0
    QDAwLCwoJCQgHBgUEAwK7BA4NnAIEBAYHCAKJCgsMDAwNDQcWDgza2traAU0FAwICAwX+swF3FQQ
    DAgEBAGMC/nL9rgIyAgABAgQFBgYDBYQHBgYFBQQAiEfCQoJCQkIBwcGBgUEAwICcAFIBQEBAgM
    DBQUGBgICAKKCgV9AgMEBgYHCAKJCwsLDA1QAgUBhA0NDAsKCgoIBwcGBQQCAGEBAGMELSA+Hz8
    CBAQB/wUDAiD+SgECAwIBwwQDAGFb/PdfA8gAAAAACAAAAAPzAyKAcQEIAAABDYAVHw41Pw01Lwo
    1LxErAQ8CKwEvCQcnDwgvBSsCDxQVDw4VHw8zLwQ1Px8fCj8ELxMPAGICCwwLCgoKCQkLCggIBgU
    EAWECEg8QDg0LCgkIBgUEAgEBAQIEBQYICQoLDQ4PDg4PAh0ODgYHBgCHBQUFAwMDBAYDBgcICQo
    MDBMCAgQCBAGFBgYGBwcHCAgJCRIUEwoSEhQCARYICw0NDQ4PDw8XFwoVExIQDw4NCQwLCwsLCws
    LCwoLCwoLCgsJCggHBwYGBQQDAwIBARAPDgwLCgkHBwYEBAMBAQECAUGBwgJCgoKCgoLCwtgCwY
    
```



```
FAwICAwQGBwkKCw0NDg8QEgIBBAMHBwgKCwWNDxITExMUFBUUDw8ODQwMCwsJCAgJERsFBAQFBgY
HCAGJCgoLDawNFBuUFhELCwKEAgMEBQUHBwgNDQ0PDw8QEgIBAgMFBgYICakKCwWOCgsKCgsKFA8
NDAsKCQcGBAMBAQEbbAMDBAYGBwcICakIGBI dCA0LCggHBgUGAhAREQkICakICAcHBgYFBAQDBQM
DBQgbBwoJCAYFAwIBAZ0DBwkLDQ4QEg8GBAQDAgICAgQEBQYGCACICQkJCgkLCgsLDAsZAwQGBgc
ICakKCwsLDawMDA0NDQsMCgoKCQcGBQQDAwEUEBAQEBEQE8PDQwLCgkJBwYFBQECDgwREA8ODgw
LCwoIBgUCAQEDBAUGBwgJCwsMBAMDawEZDQ0NDAsLCwoJCQgICAYGCAYEAgEBAGADAAAAAAPFA/Q
ACwAPABMAAAEHFzcVMzUXNyc1IyUhESENITUhAdq+NYlLiJw/S/6JAzX8y0MDwPxAAQC/NYp+foo
1v1ErAc5DZwAAAAASAN4AAQAAAAAAAAABAAAAQAAAAAAQAAEAAQAAAAAAgAHABEAAQAAAAA
AAwAQABgAAQAAAAAABAAQACgAAQAAAAAABQALADgAAQAAAAAABgQAEMAAQAAAAAACgAsAFMAAQ
AAAAACwASAH8AAwABBAkAAAACAJEAAwABBAkAAQAgAJMAAwABBAkAAgAOALMAAwABBAkAAwAgAME
AAwABBAkABAAGAOEAwABBAkABQAWAQEAwABBAkABgAgARCAAwABBAkACgBYATCAAwABBAkACwA
kAY8gdGVtcGxhdGVfdXBkYXRlZFJlZ3VsYXJ0ZW1wbGF0ZV91cGRhdGVkdGVtcGxhdGVfdXBkYXR
lZFJlZ3VsYXJ0ZW1wbGF0ZV91cGRhdGVkdGVtcGxhdGVfdXBkYXRlZFJlZ3VsYXJ0ZW1wbGF0ZV91
pb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAAdABlAG0AcABsAGEAdABlAF8AdQB
wAGQAYQB0AGUAZABSAGUAZwB1AGwAYQByAHQAZQBtAHAAbABhAHQAZQBfAHUAcABkAGEAdABlAGQ
AdABlAG0AcABsAGEAdABlAF8AdQBwAGQAYQB0AGUAZABWAGUAcBzAGkAbwBuACAAMQAuADAAAdAB
lAG0AcABsAGEAdABlAF8AdQBwAGQAYQB0AGUAZABGAG8AbgB0ACAAZwB1AG4AZQByAGEAdABlAGQ
AIAB1AHMAAQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZAB
pAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAAAAKAAAAAA
AAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgANcHJvamVjdG9yLW9sZApwb3dlcnBvaW50CG9
uZWRYaXZlDXByb2ply3RvciluZXcAAAA=) format('true');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'template' !important;
speak: none;
font-size: 40px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-projector:before { content: "\e700"; }
.sf-icon-powerpoint:before { content: "\e701"; }
.sf-icon-onedrive:before { content: "\e702"; }
.default-stepper-section {
width: 75%;
margin: 0px auto;
min-width: 85px;
padding: 25px 0;
}
}
```

Tooltip in EJ2 JavaScript Stepper control

The Stepper control supports tooltip to show additional information in the steps by setting the [showTooltip](#) property to `true`.

The tooltip appears when the user hovers over the step, providing the information such as the label or text. By default, the `showTooltip` property is `false`.

INDEX.JS

```
var iconWithLabel = [
{ label: 'Cart', iconCss: 'sf-icon-cart' },
```

```

{ label: 'Delivery Address', iconCss: 'sf-icon-transport' },
{ label: 'Payment', iconCss: 'sf-icon-payment' },
{ label: 'Confirmation', iconCss: 'sf-icon-success' }
];
var tooltipStepper = new ej.navigations.Stepper({
  steps: iconWithLabel,
  showTooltip: true
});
tooltipStepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  margin-top: 30px;
  padding: 30px;
}
@font-face {

```

```
font-family: 'Default';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIIABOCc5wbnCOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBGcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQcHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICACGBGQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwC
GBGUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICACGBQQAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBGcH/iwNDAwLCwoJCQgHBGUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAGFBgIWAQgHCPcBAGQGBGgKCgsMDQ4PDxQEBApDw4NDAsLCQgGBGQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCasMDQ4
IERESFBQUFQDQAgECAGMEBAUGBgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBGQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBGQEBAQCAQECAGwQEBAYFBwYHCAgICAgICAcIBGcFBGQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBGUFBAQCAgEBAQECAGQEBQUGBGcHBwG
ICAKIBwGHBwYGBQUEAwMCAQECAGwMEBQUGBGcHCAcICQgICACHBwYGBQUEBAICAQEBQICBAQFBQY
GBwCHCAgICQgHCAcHBGyYFBQQDAwIBAQIDAQwQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQE
BAikCAGIEAwQFDA0SBwcGAGIBAQICBgHbXyKcQkJCAcHBGUFBAQCAQEBQIDAQwQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBQEBAGYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECAGwMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXFxcWFxYXfHwYFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBGUEAgIBAQQCAQEBAwkRNRI
HBqADChI1DQoFAGEBAGMEBAOMew8eTw4IVxkXCwkJBwYCOAIBAIIDAUGBwGJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAGwMEBAUGBgYHCAgICQgHBwCGBGyYFBQQEBAQYDIgICAQECaI
CBAUGBwGJCQMBAGEMAQUDAwIDAQICBAQDBAQEBQEAwQEAGQEBQICBAMFBQUGBwCICAgJBwGHBGc
GBGUFBAQEBQqIAGeBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQwUFBgYICAKJCwsMKSckIiAeGxOYfHq
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwNdxAREXQWGBobHiAiJcCoAMDawQEA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwYAwEaBwQBAQIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYS
EJwwGAGIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDGcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAAAAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAAAADAACADwABAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAABAAAAAABAAcAwLwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAYwADAAEECQALACQBIyBEZWZhdWx0UmVndWxhcR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
```

```
1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAA
AAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueype');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }
```

Tooltip template

You can use the [tooltipTemplate](#) property to specify a custom template for the tooltips, providing detailed information about the steps.

When hovering over the step, the current step model is passed in the template context, allowing you to include dynamic information about the step.

INDEX.JS

```
var tooltipStep = [
    { text: "Review your cart items", label: 'Cart', iconCss: 'sf-icon-cart' },
    { text: "Enter your delivery address", label: 'Delivery Address', iconCss: 'sf-icon-transport' },
    { text: "Complete your purchase securely", label: 'Payment', iconCss: 'sf-icon-payment' },
    { text: "Verify your order details", label: 'Confirmation', iconCss: 'sf-icon-success' }
];
var stepperTooltip = new ej.navigations.Stepper({
    steps: tooltipStep,
    showTooltip: true,
    tooltipTemplate: '<span class="template-content"><span class="stepper-icon ${value.iconCss}"></span><span class="step-label">${value.text}</span></span>'
});
stepperTooltip.appendTo("#stepper");
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  margin-top: 30px;
  padding: 30px;
}
.template-content {
  display: flex;
  align-items: center;
}
.e-stepper-tooltip {
  border-radius: 5px;
}
.e-stepper-tooltip .e-tip-content {
  padding: 7px;
}
.step-label {
  margin-left: 7px;
  font-weight: bold;
}

```

```

}
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vSgcAAAEoAAAAVmNtYXcDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YBg9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYw1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AoAAAAACAACAAAAAABAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAQAAZABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAABAAAAAABAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAAA
EAAAABAAAAAQAQAAAAAABAAAAAAGAAAAAUAAMAAQAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAABAAAAAABAAwADAAMAawADAAAAEABAAACAAMABQAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAGIBgcFBQOEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKyp/10ICAcGBQQAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgY
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQOHCPCBAQOGBBgKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCgqFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQDBgZ
qBgOHBQMMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQAQQAFCa0LDQ8QEHlUFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECawQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAGQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECawMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgHBxYKQKJCACHBgUFBAMCAQEBAQIDAQWQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXFxcWFxYXfHwYFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwKRNRI
HBqADChI1DQoFagEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAUGBwgJCgICAQENAEFAwI
DAgECAGMFAwMEBAUDBAwMFAwIBAQECawMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAQYDIgICAQECaiI
CBAUGBwgJCQMBagEMAQUDAwIDAQICBAQDBAQEBAQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQQAiAgEBAf6RDAsLCQkICAYGBQUdAwIBAQIDAwUFBgYICAKJCwsMKSckIiAeGxoYfHq
TERAPDQwLCgkIDxsJBQUBQQAeAKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAkIBAMDawMCAQICAwYAwEaBwQBAgIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAACAABAAAAAADAACADwABAAAAAAEAACAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA

```

```

FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFTIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAbnAGUAbgB1AHIAZQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAbnAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAA
AAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');
font-weight: normal;
font-style: normal;
}

[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}

.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Animation in EJ2 JavaScript Stepper control

The Stepper progress state can be animated, smoothly transitioning from one step to another. You can customize the animation's **duration** and **delay** by using the [animation](#) property.

You can disable the animation by setting the [enable](#) property to **false**. By default, the value is **true**.

| Fields | Type | Description |

|-----|-----|-----|

| [duration](#) | **number** | Specifies the duration of the animated transition for each step. The default value is **2000** milliseconds. |

| [delay](#) | **number** | Specifies the delay to initiate the animated transition for each step in milliseconds. The default value is **0**. |

The example demonstrates the animation **duration** and **delay** settings for the Stepper.

INDEX.JS

```

var animationStepper = new ej.navigations.Stepper({
  steps: [{}, {}, {}, {} ],
  animation: {enable: true, duration: 2000, delay: 500}
});
animationStepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
  ="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Globalization in EJ2 JavaScript Stepper control

Localization

The Localization library allows you to localize the default text content of the Stepper. You can change the static text content used for the **optional** property to other cultures (Arabic, Deutsch, French, etc.) by defining the **locale** value and its translation object.

| Locale key | en-US (default) |

|-----|-----|

| optional | Optional |

In this example, the **French** culture is set to Stepper and the default text is updated with the content defined by the locale key.

INDEX.JS


```

ej.base.L10n.load({
  'fr-BE': {
    'stepper': {
      'optional': "facultatif"
    }
  }
});
var iconWithLabel = [
  { label: 'Cart', iconCss: 'sf-icon-cart' },
  { label: 'Delivery Address', iconCss: 'sf-icon-transport' },
  { label: 'Payment', iconCss: 'sf-icon-payment', optional: true },
  { label: 'Confirmation', iconCss: 'sf-icon-success' }
];
var stepper = new ej.navigations.Stepper({
  steps: iconWithLabel,
  locale: 'fr-BE'
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

STYLES.CSS

```
#container {
    margin-top: 30px;
    padding: 30px;
}
@font-face {
    font-family: 'Default';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAAAQAQAAAAEAAAAAQAQAAAAAUAAMAAQAAABQABABKAAAADAAIAAIIABOCc5wbnCOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAAwADAAAAAEABAACAAMABQAAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxdDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBGcFBQQEAgH+CwEBAGQEBQUHBGgICAK
JCgoKCQoICQgHBwCFBQQDAwEBAQEDAwQFBQcHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICACGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICACGBQQAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgC/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgyEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQgHCPcBAgQGBgGKCgsMDQ4PDxQEBAQPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDAk
IBgMBAQMGCakMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCasMDQ4
IERESFBQUFQDQAgECAGMEBAUGBgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwGICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBgQEBAMCAQECawQEBAYFBwYHCAgICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgHBwG
ICAKIBwGHBwYGBQUEAwMCAQECawMEBQUGBgHCAcICQgICACHBwYGBQUEBAICAEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAGENDAwKCggHBQE
BAikCAGIEAwQFDA0SBwcGAgIBAQICBgcHBxYKQkJCAcHBgUFBAMCAQEBAQIDAQWQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXFhCXfXcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBawKRNRI
HBqADChI1DQoFagEBAgMEBAOMew8eTw4IVxkXCwkJBwYCOAIBaIIDAUGBwGJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAQYDIgICAQECAiI
CBAUGBwGJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwCICAgJBwGHBGc
```

```

GBgUFBAQEBOQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAwUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLcGkIDxsJBQUFBQnEakKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REakIBAMDawMCAQICawcYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAAAADAACADwABAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAGAACAKAABAAAAAAAKACwAlwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFTIAZQB
nAHUAbABhAHIArABLAGYAYQBLAGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAIAYQB0AGUAZAAGAHUAacwBpAG4AZwA
gAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAA
AAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('true');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

RTL

RTL provides an option to switch the text direction and layout of the Stepper control from right to left by setting the [enableRtl](#) property to **true**.

INDEX.JS

```

var iconWithText = [
  { text: 'Cart', iconCss: 'sf-icon-cart' },
  { text: 'Delivery Address', iconCss: 'sf-icon-transport' },
  { text: 'Payment', iconCss: 'sf-icon-payment' },
  { text: 'Confirmation', iconCss: 'sf-icon-success' }
];
var stepper = new ej.navigations.Stepper({
  steps: iconWithText,
  enableRtl: true
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  margin-top: 30px;
  padding: 30px;
}
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAAA
EAAAABAAAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAA

```

```

AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQCAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgcH/iwNDawLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHW8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CfHITE+WDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEa5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAqQHCPCBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKcAYGBAL+KgEEBQgKcW0PEBETFBQWFxcXfHfYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERUMFBYWFxcXfHfQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCasMDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKcAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDByVFRQSERAPDQsKCAUEAQEEBQgKcW0
PEBESFBUVFhcXfXyVFBiSEA8NCwoIBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAWCAQECaWQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECaWMEBQUGBgcHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8CowEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BagoCAGENDAwKCgghBQE
BAikCAGIEAwQFDA0SBwcGAgIBAQICBgcHBxYKQkKJCAcHBgUFBAMCAQEBAQIDAQWFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYQFBAQEBAgYHBwcWCgkKCAgYHwYFBQQAQAgE
BAQECAwMEBAyFBgcHEBABAQED/qwUFRUVFRYWFHfYWFxYXfHcXfXcWFxYXfHfYWFHfYFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqYVGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqYCAQEBAwKRNRI
HBqADChI1DQoFAGeBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAQWUGBwgJCgICAQENAEQFAwI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQIAGeBAf6RDAsLCQkICAYGBQUdAwIBAQIDAQWFBgYICakJCwsMKsckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJcCoAMDawQECA8XPRcKCGUPFz0
REAkIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAGIwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAACAAQABAAAAAACAACACAABAAAAAADAACADwABAAAAAAEAACAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZwZhdWx0UmVndWxhckR
lZmFlbHREZwZhdWx0VmVyc2lubiAxLjBEZwZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lubiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8ABQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAA
AAAAAAYBAGEDAQQBBQEGAQCADXRYW5zcG9ydC12YW4ldXNlciltb2RpZnRlc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
    font-family: 'Default' !important;

```

```

speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Accessibility in EJ2 JavaScript Stepper control

Accessibility is achieved in the Stepper control through WAI-ARIA standard and keyboard navigations. The Stepper control can be effectively accessed through assistive technologies such as screen readers.

Keyboard interaction

The Stepper control is interactive with the below keyboard shortcuts.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Up Arrow | Focuses the previous step in a vertical Stepper. |

| Down Arrow | Focuses the next step in a vertical Stepper. |

| Left Arrow | Focuses the previous step in a horizontal Stepper. |

| Right Arrow | Focuses the next step in a horizontal Stepper. |

| Home | Focuses on the first step of the Stepper. |

| End | Focuses on the last step of the Stepper. |

| Enter / Space | Activates the currently focused step. |

ARIA attribute

The following ARIA attributes are used in Stepper control based on its state.

| Properties | Functionality |

| ----- | ----- |

| **aria-label** | Attribute provide a descriptive text that labels the interactive element for accessibility. |

| **aria-current** | Attribute denotes the currently active step in the Stepper. |

| **aria-disabled** | Indicates when the step is disabled and cannot be interacted. |

Events in EJ2 JavaScript Stepper control

This section describes the Stepper events that will be triggered when an appropriate actions are performed. The following events are available in the Stepper control.

created

The Stepper control triggers the [created](#) event when the control rendering is completed.

INDEX.JS

```

var stepper = new ej.navigations.Stepper({
  steps: [{}, {}, {}, {}],
  created: () => {
    //Your required action here
  }
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

stepChanged

The Stepper control triggers the [stepChanged](#) event after the active step is changed.

INDEX.JS

```

var stepper = new ej.navigations.Stepper({
  steps: [{}, {}, {}, {}],

```

```

    stepChanged: (args) =>{
        alert("Step changed from "+args.previousStep + " to " + args.activeStep)
    }
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

stepChanging

The Stepper control triggers the [stepChanging](#) event before the active step change.

INDEX.JS

```

var stepper = new ej.navigations.Stepper({
  steps: [{}, {}, {}, {}],
  stepChanging: (args) =>{
    alert("Step changing from "+args.previousStep + " to " +
args.activeStep)
  }
}

```



```
});
stepper.appendTo("#stepper");
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

stepClick

The Stepper control triggers the [stepClick](#) event when the step is clicked.

INDEX.JS

```
var stepper = new ej.navigations.Stepper({
  steps: [{}, {}, {}, {}],
  stepClick: (args) =>{
    //Your required action here
  }
});
stepper.appendTo("#stepper");
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
  <meta name="description" content="Essential JS 2">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
  <!--style reference from app-->
  <link href="styles.css" rel="stylesheet">
  <!--system js reference and configuration-->

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <nav id="stepper"></nav>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

beforeStepRender

The Stepper control triggers the [beforeStepRender](#) event before rendering each step.

INDEX.JS

```

var stepper = new ej.navigations.Stepper({
  steps: [{}, {}, {}, {}],
  beforeStepRender: (args) =>{
    //Your required action here
  }
});
stepper.appendTo("#stepper");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 - Stepper</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no">
<meta name="description" content="Essential JS 2">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<!--style reference from app-->
<link href="styles.css" rel="stylesheet">
<!--system js reference and configuration-->

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <nav id="stepper"></nav>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stock Chart

<!-- markdownlint-disable MD036 -->

Working with data in EJ2 JavaScript Stock chart control

Stock Chart can visualise data bound from local or remote data.

Local Data

You can bind a simple JSON data to the chart using [dataSource](#) property in series.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, HiloOpenCloseSeries, CandleSeries } from
 '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair}

```

```

    from '@syncfusion/ej2-charts';
    StockChart.Inject(DateTime, HiloOpenCloseSeries, CandleSeries);
    StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
    BollingerBands, EmaIndicator, MomentumIndicator);
    StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
    StochasticIndicator);
    StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
    Crosshair, Export);
    let stockChart: StockChart = new StockChart({
        series: [
            {
                dataSource: chartData, type: 'Candle',
                animation: { enable: true },
                bearFillColor: '#2ecd71', bullFillColor: '#e74c3d',
            }
        ],
    });
    stockChart.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

    <script src="./systemjs.config.js"></script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Series Types](#)

Chart dimensions in EJ2 JavaScript Stock chart control

Size for Container

Stock Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, CandleSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair}
    from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, CandleSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ]
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Chart dimension</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="./material.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
```

```

        <div id="element"></div>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Size for Stock Chart

<!-- markdownlint-disable MD036 -->

You can also set size for chart directly through [width](#) and [height](#) properties.

In Pixel

You can set the size of chart in pixel as demonstrated below.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, CandleSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair}
    from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, CandleSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
    // Width and height for stock chart in pixel
    width: '650', height: '350'
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Chart dimension</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="./index.css" rel="stylesheet">
<link href="./material.css" rel="stylesheet">
<link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

In Percentage

By setting value in percentage, Stock Chart gets its dimension with respect to its container. For example, when the height is '50%', Stock Chart renders to half of the container height.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, CandleSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair}
    from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, CandleSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    series: [
        {
            dataSource: chartData,

```

```

        type: 'Candle'
    },
],
// Width and height for stock chart in percentage
width: '80%', height: '90%'
});
stockChart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Chart dimension</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="./material.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: When you do not specify the size, it takes **450px** as the height and window size as its width.

<!-- markdownlint-disable MD036 -->

Axis types in EJ2 JavaScript Stock chart control

DateTime axis

DateTime axis uses date time scale and displays the date time values as axis labels in the specified format. To use DateTime axis, set the [valueType](#) of axis to **DateTime**.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
```



```
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair }
  from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
  primaryXAxis: { valueType: 'DateTime' },
  series: [
    {
      dataSource: chartData,
      type: 'Candle'
    },
  ],
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use this axis, we need to inject `DateTime` using `StockChart.Inject(DateTime)` method and set the [valueType](#) of axis to `DateTime`.

`DateTimeCategory` axis

`DateTimeCategory` axis in the stock chart is used to display only business days. To use `DateTimeCategory` axis, set the [valueType](#) of axis to `DateTimeCategory`.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTimeCategory, DateTime, AreaSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, LineSeries, SplineSeries } from
 '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair }
  from '@syncfusion/ej2-charts';
StockChart.Inject(DateTimeCategory, DateTime, AreaSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
interface DataPoint {
  x: Date;
}
let datetimeCategoryData: DataPoint[] = [
  { x: new Date(2021, 1, 11) }, { x: new Date(2021, 1, 12) }, { x: new
Date(2021, 1, 13) }, { x: new Date(2021, 1, 14) }, { x: new Date(2021, 1,
15) },
  { x: new Date(2021, 1, 19) }, { x: new Date(2021, 1, 20) }, { x: new
Date(2021, 1, 21) }, { x: new Date(2021, 1, 22) }, { x: new Date(2021, 3, 1)
},
  { x: new Date(2021, 3, 2) }, { x: new Date(2021, 4, 1) }, { x: new
Date(2021, 4, 5) }, { x: new Date(2021, 4, 6) }, { x: new Date(2021, 4, 7)
},
  { x: new Date(2021, 4, 11) }, { x: new Date(2021, 4, 13) }, { x: new
Date(2021, 4, 15) }, { x: new Date(2021, 4, 16) }, { x: new Date(2021, 4,
17) },
  { x: new Date(2021, 4, 18) }, { x: new Date(2021, 4, 20) }, { x: new
Date(2021, 4, 21) }, { x: new Date(2021, 4, 23) }, { x: new Date(2021, 4,
25) },
  { x: new Date(2021, 5, 1) }, { x: new Date(2021, 5, 2) }, { x: new
Date(2021, 5, 6) }, { x: new Date(2021, 5, 7) }, { x: new Date(2021, 5, 8)
},
  { x: new Date(2021, 5, 11) }, { x: new Date(2021, 5, 15) }, { x: new
Date(2021, 5, 18) }, { x: new Date(2021, 5, 20) }, { x: new Date(2021, 5,
25) },

```

```

    { x: new Date(2021, 6, 1) }, { x: new Date(2021, 6, 2) }, { x: new
Date(2021, 6, 3) }, { x: new Date(2021, 6, 4) }, { x: new Date(2021, 6, 5)
},
    { x: new Date(2021, 6, 10) }, { x: new Date(2021, 6, 11) }, { x: new
Date(2021, 6, 12) }, { x: new Date(2021, 6, 13) }, { x: new Date(2021, 6,
15) },
    { x: new Date(2021, 6, 16) }, { x: new Date(2021, 6, 17) }, { x: new
Date(2021, 6, 18) }, { x: new Date(2021, 6, 19) }, { x: new Date(2021, 6,
20) }
]
let series2: Object[] = [];
let point2: Object;
for (var i = 0; i < 46; i++) {
    point2 = {
        x: datetimeCategoryData[i].x,
        y: getRandomInRange(120, 130),
        high: getRandomInRange(88, 92),
        low: getRandomInRange(76, 86),
        open: getRandomInRange(75, 85),
        close: getRandomInRange(85, 90),
        volume: getRandomInRange(660187068, 965935749)
    };
    series2.push(point2);
}
function getRandomInRange(min: number, max: number): number {
    const randomDecimal = Math.random();
    const randomValue = randomDecimal * (max - min) + min;
    return randomValue;
}
let stockChart: StockChart = new StockChart({
    primaryXAxis: { valueType: 'DateTimeCategory', majorGridLines: { width: 0
}}, crosshairTooltip: { enable: true } },
    series: [
        {
            dataSource: series2,
            type: 'Line', xName: 'x', yName: 'y'
        },
    ],
    tooltip: {
        enable: true, header: 'AAPL Stock Price'
    },
    crosshair: {
        enable: true
    },
});
stockChart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
```

```

<link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use this axis, we need to inject `DateTimeCategory` using `StockChart.Inject(DateTimeCategory)` method and set the `valueType` of axis to `DateTimeCategory`.

Logarithmic axis

<!-- markdownlint-disable MD033 -->

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numerical values in both lower order of magnitude (eg: 10^{-6}) and higher order of magnitude (eg: 10^6). To use Logarithmic axis, set the `valueType` of axis to `Logarithmic`.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries, HiloSeries,
LineSeries, Logarithmic } from '@syncfusion/ej2-charts';
import { SplineAreaSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
IStockChartEventArgs, ChartTheme } from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, Logarithmic, AreaSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, LineSeries, SplineAreaSeries, SplineSeries
);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);

```

```

let stockChart: StockChart = new StockChart({
  primaryXAxis: { valueType: 'DateTime' },
  primaryYAxis: {
    valueType: 'Logarithmic'
  },
  chartArea: { border: { width: 0 } },
  series: [
    {
      dataSource: chartData, type: 'Line' xName: 'date' yName:
'high'
    }
  ],
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use this axis, we need to inject Logarithmic using `StockChart.Inject(Logarithmic)` method and set the `valueType` of axis to `Logarithmic`.

See also

- [Axis Customization](#)

Axis customization in EJ2 JavaScript Stock chart control

Axis Crossing

An axis can be positioned in the Stock Chart area using [crossesAt](#) and [crossesInAxis](#) (`../api/stock-chart/stockChartAxisModel/#crossesinaxis`) properties. The [crossesAt](#) property specifies the values (datetime, numeric, or logarithmic) at which the axis line has to be intersected with the vertical axis or vice-versa, and the [crossesInAxis](#) property specifies the axis name with which the axis line has to be crossed.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair }
    from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        crossesAt: 90
    },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Title

You can add a title to the axis using [title](#) property to provide quick information to the user about the data plotted in the axis. Title style can be customized using [titleStyle](#) property of the axis.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair }
    from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        title: 'AAPL Historical',
    },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tick Lines Customization

You can customize the **width**, **color** and **size** of the minor and major tick lines, using [majorTickLines](#) and [minorTickLines](#) properties in the axis.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs }
  from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);

```



```

StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
  primaryXAxis: {
    //Tick lines customization
    majorTickLines: {
      color: 'blue',
      width: 5
    },
    minorTickLines: {
      color: 'red',
      width: 0
    }
  },
  primaryYAxis: {
    //Grid lines customization
    majorTickLines: {
      color: 'green',
      width: 5
    },
    minorTickLines: {
      color: 'red',
      width: 0
    }
  },
  series: [
    {
      dataSource: chartData,
      type: 'Candle'
    },
  ],
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

    <div id="container">
      <div id="element"></div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Grid Lines Customization

You can customize the **width**, **color** and **dashArray** of the minor and major grid lines, using [majorGridLines](#) and [minorGridLines](#) properties in the axis.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs }
  from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
  primaryXAxis: {
    //Grid lines customization
    majorGridLines : {
      color : 'blue',
      width : 1
    },
    minorGridLines : {
      color : 'red',
      width : 0
    }
  },
  primaryYAxis: {
    //Grid lines customization
    majorGridLines : {
      color : 'green',
      width : 1
    },

```

```

        minorGridLines : {
            color : 'red',
            width : 0
        },
    },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
});
stockChart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Multiple Axis

In addition to primary X and Y axis, we can add n number of axis to the Stock Chart. Series can be associated with this [axis](#), by mapping with axis's unique name.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
```

```

import { DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair}
  from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, LineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
  series: [
    {
      dataSource: chartData, type: 'Line', xName: 'date' yName:
'high'
    },
    {
      dataSource: chartData, type: 'Line', xName: 'date' yName:
'low'
      yAxisName: 'yAxis',
    }
  ],
  // Initializing multiple axis
  axes:[
    {
      rowIndex: 0,
      name: 'yAxis',
    }
  ],
  crosshair: {
    enable: true
  },
  title: 'Multiple Axis',
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Inversed Axis

<!-- markdownlint-disable MD033 -->

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner set this property

[isInversed](#) to true.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    //Initializing Primary X Axis
    primaryXAxis: {
        isInversed: true
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        isInversed: true
    },

```

```

series: [
  {
    dataSource: chartData,
    type: 'Candle'
  },
],
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Opposed Position

To place an axis opposite from its original position, set [opposedPosition](#) property of the axis to true.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';

```

```

import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs }
  from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
  primaryXAxis:
  {
    opposedPosition: true
  },
  series: [
    {
      dataSource: chartData,
      type: 'Candle'
    },
  ],
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Series types in EJ2 JavaScript Stock chart control

Essential JS 2 StockChart supports 6 major types of series namely **Line**, **Spline**, **Hilo**, **HiloOpenClose**, **Hollow Candle** and **Candle**. By using the series dropdown button you can navigate between the above listed series types.

```
<!-- markdownlint-disable MD036 -->
```

Line

To render a line series, use series **type** as **Line** and inject **LineSeries** module using **StockChart.Inject(LineSeries)** method.

Spline

To render a spline series, use series **type** as **Spline** and inject **SplineSeries** module using **StockChart.Inject(SplineSeries)** method.

Hilo

To render a hilo series, use series **type** as **Hilo** and inject **HiloSeries** module using **StockChart.Inject(HiloSeries)** method.

HiloOpenClose

To render a hiloOpenClose series, use series **type** as **HiloOpenClose** and inject **HiloOpenCloseSeries** module using **StockChart.Inject(HiloOpenCloseSeries)** method.

HollowCandle

To render a hollowcandle series, use series **type** as **Candle** and set **enableSolidCandle** as false. Now inject **CandleSeries** module using **StockChart.Inject(CandleSeries)** method.

Candle

To render a candle series, use series **type** as **Candle** and inject **CandleSeries** module using **StockChart.Inject(CandleSeries)** method.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { RangeTooltip, Tooltip, Crosshair }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(RangeTooltip, Tooltip, Crosshair);
let stockChart: StockChart = new StockChart({
series:[{
    dataSource: chartData,
    type: 'Candle'
}],
indicatorType: [];
trendlineType: [];
exportType: [];
});
```



```
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

```
<!-- markdownlint-disable MD036 -->
```

Trend lines in EJ2 JavaScript Stock chart control

Trendlines are used to show the direction and speed of price.

StockChart supports 6 types of trendlines namely

Linear, Exponential, Logarithmic, Polynomial, Power, Moving Average. By using trendline dropdown button you can add or remove the required trendline type.

Linear

A linear trendline is a best fit straight line that is used with simpler data sets. To render a linear trendline, use trendline **type** as **Linear** and inject **Trendlines** module using **StockChart.Inject(Trendlines)**.

Exponential

An exponential trendline is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline, if your data contains zero or negative values.

To render a exponential trendline, use trendline **type** as **Exponential** and inject

Trendlines module using `StockChart.Inject(Trendlines)`.

Logarithmic

A logarithmic trendline is a best-fit curved line that is most useful when the rate of change in the data increases or decreases quickly and then levels out. A logarithmic trendline can use negative and/or positive values.

To render a logarithmic trendline, use trendline [type](#) as **Logarithmic** and inject

Trendlines module using `StockChart.Inject(Trendlines)`.

Polynomial

A polynomial trendline is a curved line that is used when data fluctuates.

To render a polynomial trendline, use trendline [type](#) as **Polynomial** and inject

Trendlines module using `StockChart.Inject(Trendlines)`.

polynomialOrder used to define the polynomial value.

Power

A power trendline is a curved line that is best used with data sets that compare measurements that increase at a specific rate.

To render a power trendline, use trendline [type](#) as **Power** and inject **Trendlines** module using `StockChart.Inject(Trendlines)`.

Moving Average

A moving average trendline smoothen out fluctuations in data to show a pattern or trend more clearly.

To render a moving average trendline, use trendline [type](#) as **MovingAverage** and inject

Trendlines module using `StockChart.Inject(Trendlines)`.

period property defines the period to find the moving average.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries, HiloSeries,
LineSeries, Logarithmic } from '@syncfusion/ej2-charts';
import { SplineAreaSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
IStockChartEventArgs, ChartTheme } from '@syncfusion/ej2-charts';
import { chartData } from '../datasource.ts';
StockChart.Inject(DateTime, Logarithmic, AreaSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, LineSeries, SplineAreaSeries, SplineSeries
);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
```

```

let stockChart: StockChart = new StockChart({
  primaryXAxis: {
    title: 'Months',
  },
  primaryYAxis: {
    title: 'Rupees against Dollars',
    interval: 5
  },
  tooltip: { enable: true },
  chartArea: { border: { width: 0 } },
  series: [{
    dataSource: chartData,
    name: 'Apple Inc',
    fill: '#0066FF',
    //Series type as candle
    type: 'Candle',
    trendlines: [{ type: 'MovingAverage', enableTooltip: false } ]
  }],
  seriesType: [],
  indicatorType: [],
  exportType: [],
  title: 'Historical Indian Rupee Rate (INR USD) '
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

Customization of Trendline

The [fill](#) and [width](#) properties are used to customize the appearance of the trendline.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries, HiloSeries,
LineSeries, Logarithmic } from '@syncfusion/ej2-charts';
import { SplineAreaSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
IStockChartEventArgs, ChartTheme } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
StockChart.Inject(DateTime, Logarithmic, AreaSeries, CandleSeries,
HiloOpenCloseSeries, HiloSeries, LineSeries, SplineAreaSeries, SplineSeries
);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        title: 'Months',
    },
    primaryYAxis: {
        title: 'Rupees against Dollars',
        interval: 5
    },
    tooltip: { enable: true },
    chartArea: { border: { width: 0 } },
    series: [{
        dataSource: chartData,
        name: 'Apple Inc',
        fill: '#0066FF',
        //Series type as candle
        type: 'Candle',
        trendlines: [{ type: 'MovingAverage', enableTooltip: false, fill:
'red', width: 2 }]
    }],
    seriesType: [],
    indicatorType: [],
    exportType: [],
    title: 'Historical Indian Rupee Rate (INR USD)'
});
stockChart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Technical indicators in EJ2 JavaScript Stock chart control

A technical indicator is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

StockChart supports 10 types of technical indicators namely Accumulation Distribution, ATR, EMA, SMA, TMA, Momentum, MACD, RSI, Stochastic, Bollinger Band. By using indicator dropdown box you can add and remove the required indicators types.

Accumulation Distribution

Accumulation Distribution combines price and volume to show how money may be flowing into or out of stock. To render a Accumulation Distribution Indicator, use indicator [type](#) as AccumulationDistribution and inject

AccumulationDistributionIndicator module using StockChart.Inject(AccumulationDistributionIndicator). To calculate the signal line [volume](#) field is additionally added with [dataSource](#).

Average True Range (ATR)

ATR measures the stock volatility by comparing the current value with the previous value. To render a Average True Range (ATR) Indicator,

use indicator [type](#) as `Atr` and inject `AtrIndicator` module using `StockChart.Inject(AtrIndicator)`.

Exponential Moving Average (EMA)

Moving average Indicators are used to define the direction of the trend. To render a EMA Indicator, use indicator [type](#) as `Ema` and inject `EmaIndicator` module using `StockChart.Inject(EmaIndicator)`.

Momentum

Momentum shows the speed at which the price of the stock is changing. To render a Momentum indicator, use indicator [type](#) as `Momentum` and inject `MomentumIndicator` module using `StockChart.Inject(MomentumIndicator)` method. Momentum indicator will be represented by two lines (upperLine, signalLine). In momentum indicator the upperBand value is always renders at the value 100.

Moving Average Convergence Divergence (MACD)

MACD is based on the difference between two EMA's. To render a MACD Indicator, use indicator [type](#) as `Macd` and inject `MacdIndicator` module using `StockChart.Inject(MacdIndicator)`. MACD indicator will be represented by MACD line, signal line, MACD histogram. MACD histogram is used to differentiate MACD line and signal line.

Relative Strength Index (RSI)

RSI shows how strongly a stock is moving in its current direction. To render a RSI Indicator, use indicator [type](#) as `Rsi` and inject `RsiIndicator` module using `StockChart.Inject(Rsindicator)`. RSI indicator will be represented

by three lines (upperBand, lowerBand, signalLine). The upperBand and lowerBand values are customized by [overBought](#) and [overSold](#) properties of indicator and the signalLine is calculated by RSI formula.

Simple Moving Average (SMA)

Moving average Indicators are used to define the direction of the trend. To render a SMA Indicator, use indicator [type](#) as `Sma` and inject `SmaIndicator` module using `StockChart.Inject(SmaIndicator)`.

Stochastic

It shows how a stock is, when compared to previous state. To render a Stochastic indicator, use indicator [type](#) as `Stochastic` and inject `StochasticIndicator` module using `StockChart.Inject(StochasticIndicator)` method.

stochastic indicator will be represented by four lines (upperLine, lowerLine, periodLine, signalLine). In stochastic indicator the upperBand value and lowerBand value is customized by [overBought](#) and [overSold](#) properties of indicators and the periodLine and signalLine is render based on stochastic formula.

Triangular Moving Average (TMA)

Moving average Indicators are used to define the direction of the trend. To render a TMA Indicator, use indicator [type](#) as `Tma` and inject `TmaIndicator` module using `StockChart.Inject(TmaIndicator)`.

Bollinger Band

A StockChart overlay that shows the upper and lower limits of normal price movements based on the standard deviation of prices. To render a Bollinger Band, use indicator [type](#) as `BollingerBand` and inject `BollingerBands` module using `StockChart.Inject(BollingerBands)` method. Bollinger band will be represented by three lines (upperLine, lowerLine, signalLine).

The default values of the Bollinger Band [period](#) is 14 and [standardDeviations](#) is 2.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
    from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair);
let stockChart: StockChart = new StockChart({
    chartArea: { border: { width: 0 } },
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
    },
    border : {width : 0},
    primaryXAxis: { majorGridLines: { color: 'transparent' } },
    crosshairTooltip: { enable: true } },
    series: [
        {
            dataSource: chartData, name: 'Apple Inc',
            type: 'Candle', bearFillColor: '#00226C', bullFillColor:
"#0450C2", fill: 'blue'
        },
    ],
    indicators: [{
        type: 'BollingerBands', field: 'Close', seriesName: 'Apple Inc',
        xName: 'date', high: 'high', low: 'low', open: 'open', close: 'close',
        period: 10, upperLine: { color: '#ffb735', width: 1 },
        lowerLine: { color: '#f2ec2f', width: 1 }
    }],
    tooltipRender: (args: ITooltipRenderEventArgs) => {
        if (args.text.split('<br/>')[4]) {
            let target: number =
parseFloat(args.text.split('<br/>')[4].split('<b>')[1].split('</b>')[0]);
            let value: string = (target / 100000000).toFixed(1) + 'B';
            args.text =
args.text.replace(args.text.split('<br/>')[4].split('<b>')[1].split('</b>')[
0], value);
        }
    },
    seriesType: [],
    exportType: [],
    trendlineType: [],
    tooltip: {

```

```

        enable: true
    },
    crosshair: {
        enable: true
    },
    height: '350',
});
stockChart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Tool tip in EJ2 JavaScript Stock chart control

<!-- markdownlint-disable MD036 -->

StockChart will display details about the points through tooltip, when the mouse is moved over the point.

Default tooltip

By default, tooltip is not visible. Enable the tooltip by setting [enable](#) property to true and by injecting [Tooltip](#) module using `StockChart.Inject(Tooltip)`.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
```



```

import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    series: [
        {
            type: 'Candle',
            dataSource: chartData
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    //Default tooltip for StockChart
    tooltip: {enable: true}
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>

```

```

    </div>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD013 -->

Format the tooltip

<!-- markdownlint-disable MD013 -->

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format `${series.name} ${point.x}` shows series name and point x value.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    series: [
        {
            type: 'Candle',
            dataSource: chartData
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {
        enable: true,
        //tooltip format for StockChart
        header: 'Unemployment',
        format: '<b>${point.x} : ${point.y}</b>'
    }
}

```

```
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Position the tooltip

By default, the tooltip is positioned at the left side of the stock chart. You can move the tooltip along with the mouse by setting **Nearest** to the [position](#) property.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
  from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
```

```

StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    series: [
        {
            type: 'Candle',
            dataSource: chartData
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    //Default tooltip for StockChart
    tooltip: { enable: true, shared: true, position: 'Nearest' }
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the appearance of the tooltip

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    series: [
        {
            type: 'Candle',
            dataSource: chartData
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {
        enable: true,
        format: ' ${point.x} : ${point.y}',
        //fill for tooltip
        fill: '#7bb4eb',
        //border for tooltip
        border: {
            width: 2,
            color: 'grey'
        }
    }
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="./index.css" rel="stylesheet">
<link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Cross hair in EJ2 JavaScript Stock chart control

Crosshair has a vertical and horizontal line to view the value of the axis at mouse or touch position.

Crosshair lines can be enabled by using [enable](#) property in the `crosshair`.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair } from '@syncfusion/ej2-
charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },

```

```

    },
    primaryXAxis: { majorGridLines: { color: 'transparent' } },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        }
    ],
    crosshair: {
        enable: true
    },
    title: 'AAPL Stock Price'
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tooltip for axis

Tooltip label for an axis can be enabled by using [enable](#) property of `crosshairTooltip` in the corresponding axis.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
```

```

import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair } from
 '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
        crosshairTooltip: {enable:true}
    },
    primaryXAxis: { majorGridLines: { color: 'transparent' },
        crosshairTooltip: {enable:true}},
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        }
    ],
    crosshair: {
        enable: true
    },
    title: 'AAPL Stock Price'
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```



```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

The [fill](#) and [textStyle](#) property of the `crosshairTooltip` is used to customize the background color and font style of the crosshair label respectively. Color and width of the crosshair line can be customized by using the [line](#) property in the crosshair.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair } from '@syncfusion/ej2-
charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
        crosshairTooltip: { enable: true, fill: 'green' }
    },
    primaryXAxis: { majorGridLines: { color: 'transparent' },
        crosshairTooltip: { enable: true, fill: 'green' } },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        }
    ],
    crosshair: {
        enable: true,

```

```

        line: {width: 2, color: 'green'}
    },
    title: 'AAPL Stock Price'
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Note: To use crosshair feature, we need to inject `Crosshair` module `StockChart.Inject(Crosshair)` method.

```
<!-- markdownlint-disable MD036 -->
```

Legend in EJ2 JavaScript Stock chart control

```
<!-- markdownlint-disable MD038 -->
```

Legend provides information about the series rendered in the Stock Chart. Legend can be added to a Stock Chart by enabling the [visible](#) option in the [legendSettings](#).

Position and Alignment

By using the [position](#) property, legend can be placed at `Left`, `Right`, `Top`, `Bottom` or `Custom` of the Stock Chart. The legend is positioned at the bottom of the Stock Chart, by default.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    indicatorType : [],
    trendlineType : [],
    series: [
        {
            type: 'Candle',
            dataSource: chartData,
            name: 'China'
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {enable: true},
    legendSettings: {
        visible: true,
        //Legend position as top
        position: 'Top'
    }
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

[Custom](#) position is used to position the legend anywhere in the Stock Chart using x, y coordinates.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    indicatorType : [],
    trendlineType : [],
    series: [
        {
            type: 'Candle',
            dataSource: chartData,
            name: 'China'
        }
    ]
});

```

```

        title: 'Unemployment Rates 1975-2010',
        tooltip: {enable: true},
        legendSettings: {
            visible: true,
            //Legend position as custom
            position: 'Custom',
            location: { x: 200, y: 20 } }
    });
    stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
    <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Alignment

The legend can be align as **Center**, **Far** or **Near** to the Stock Chart using [alignment](#) property.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';

```

```

import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    indicatorType : [],
    trendlineType : [],
    series: [
        {
            type: 'Candle',
            dataSource: chartData,
            name: 'China'
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {enable: true},
    legendSettings: {
        visible: true,
        position: 'Bottom',
        //Legend alignment as near
        alignment: 'Near'
    }
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customization

To change the legend icon shape, [legendShape](#) property in the [series](#) can be used. By default legend icon shape is `seriesType`.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    indicatorType : [],
    trendlineType : [],
    series: [
        {
            type: 'Candle',
            dataSource: chartData,
            name: 'China',
            legendShape: 'Pentagon'
        }
    ],

```

```

        title: 'Unemployment Rates 1975-2010',
        tooltip: { enable: true },
        legendSettings: { visible: true }
    });
    stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Size

By default, legend takes 20% - 25% of the Stock Chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the width vertically, while placing on left or right position of the Stock Chart. The default legend size can be changed by using the [width](#) and [height](#) property of the [legendSettings](#).

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';

```



```

import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    indicatorType : [],
    trendlineType : [],
    series: [
        {
            type: 'Candle',
            dataSource: chartData,
            name: 'China'
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {enable: true},
    legendSettings: {
        visible: true,
        //Legend size for chart
        width: '500', height: '50',
        border: { width: 1, color: 'pink'}
    }
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Item Size

The size of the legend items can customized by using the [shapeHeight](#) and [shapeWidth](#) property.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    indicatorType : [],
    trendlineType : [],
    series: [
        {
            type: 'Candle',
            dataSource: chartData,
            name: 'China'
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {enable: true},

```

```

        legendSettings: {
            visible: true,
            //Legend item size for chart
            shapeHeight: 15, shapeWidth: 15
        }
    });
    stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Collapsing Legend Item

By default, series name will be displayed as legend. To skip the legend for a particular series, empty string to the series name can be given.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';

```

```

import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    indicatorType : [],
    trendlineType : [],
    series: [
        {
            type: 'Candle',
            dataSource: chartData
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {enable: true},
    legendSettings: { visible: true }
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Title

The title for legend can be set using [title](#) property in [legendSettings](#). Customize the [fontStyle](#), [size](#), [fontWeight](#), [color](#), [textAlignment](#), [fontFamily](#), [opacity](#) and [textOverflow](#) of legend title. [titlePosition](#) is used to set the legend position in [Top](#), [Left](#) and [Right](#) position. [maximumTitleWidth](#) is used to set the width of the legend title. By default, it will be [100px](#).

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export, StockLegend } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
ITooltipRenderEventArgs, IStockChartEventArgs, ChartTheme }
from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export, StockLegend);
let stockChart: StockChart = new StockChart({
  primaryXAxis: {
    valueType: 'DateTime'
  },
  indicatorType : [],
  trendlineType : [],
  series: [
    {
      type: 'Candle',
      dataSource: chartData,
      name: 'China'
    }
  ],
  title: 'Unemployment Rates 1975-2010',
  tooltip: {enable: true},
  legendSettings: {
    visible: true,

```

```

        title: 'Countries',
        titlePosition: 'Top',
        titleStyle: {
            fontFamily: 'verdana',
            fontStyle: 'Normal',
            fontWeight: 'Normal',
            size: '15px',
            textAlign: 'Center',
            color: 'blue',
            textOverflow: 'None'
        },
        maximumTitleWidth: 150
    }
});
stockChart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note: To use legend feature, we need to inject `StockLegend` using `StockChart.Inject(StockLegend)`.

Period selector in EJ2 JavaScript Stock chart control

The period selector allows to select a range with specified periods. By default the period selector is enabled in stock chart.

Periods

Periods is an array of objects that allows users to specify the range of [periods](#). The `interval` property specifies the count value of the button, and the `text` property specifies the text to be displayed on button. The `intervalType` property allows users to customize the intervals of the buttons. The `intervalType` property supports the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, Export, SplineSeries } from '@syncfusion/ej2-charts';
import { RangeTooltip, Crosshair } from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(RangeTooltip, Crosshair, Export);
let stockChart: StockChart = new StockChart({
    primaryXAxis: { valueType: 'DateTime', majorGridLines: { color:
'transparent' } },
    crosshairTooltip: { enable: true } },
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
        crosshairTooltip: { enable: true }
    },
    series: [
        {
            dataSource: GetDateTimeData(), xName: 'x', yName: 'y', type:
'Line', name: 'google',
        }
    ],
    seriesType: [],
    indicatorType: [],
    exportType: [],
    trendlineType: [],
    periods: [
        { intervalType: 'Minutes', interval: 1, text: '1m' },
        { intervalType: 'Minutes', interval: 30, text: '30m' },
        { intervalType: 'Hours', interval: 1, text: '1H' },
    ]
});
```

```

        { intervalType: 'Hours', interval: 12, text: '12H', selected:
true },
        { text: '1D' }
    ],
    title: 'AAPL stock price by minutes',
    crosshair: {
        enable: true
    },
});
stockChart.appendTo('#element');
function GetDateTimeData() : {x: Date, y: number}[] {
    let series1: Object[] = [];
    let point1: Object;
    let value: number = 80;
    let i: number;
    for (i = 1; i < 500; i++) {
        if (Math.random() > .5) {
            value += Math.random();
        } else {
            value -= Math.random();
        }
        point1 = { x: new Date(2000, 1, 1, 0, i), y: value.toFixed(1) };
        series1.push(point1);
    }
    return series1;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="es5-datasource.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>

```



```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Visibility of period selector

The [enablePeriodSelector](#) property allows users to toggle the visibility of period selector.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair } from '@syncfusion/ej2-
charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    enablePeriodSelector: false,
    primaryXAxis: { valueType: 'DateTime', majorGridLines: { color:
'transparent' },
        crosshairTooltip: { enable: true } },
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
        crosshairTooltip: { enable: true }
    },
    series: [
        {
            dataSource: chartData, type: 'Candle', name: 'google',
        }
    ],
    title: 'AAPL stock price by minutes',
    crosshair: {
        enable: true
    }
});
stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">

```

```

<meta name="author" content="Syncfusion">
<link href="./index.css" rel="stylesheet">
<link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Range selector in EJ2 JavaScript Stock chart control

The left and right thumb of RangeNavigator are used to indicate the selected range in the large collection of data. Following are the ways you can select a range.

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and end through Date Range button.

Following code example shows the [enableSelector](#) property allows users to toggle the visibility of range selector.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair } from '@syncfusion/ej2-
charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);

```

```
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
  enableSelector : false,
  primaryYAxis: {
    lineStyle: { color: 'transparent' },
    majorTickLines: { color: 'transparent', width: 0 },
  },
  primaryXAxis: { majorGridLines: { color: 'transparent' } },
  series: [
    {
      dataSource: chartData,
      type: 'Candle'
    },
  ],
  title: 'AAPL Stock Price'
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Export print in EJ2 JavaScript Stock chart control

The rendered stock chart can be exported to JPEG, PNG, SVG, or PDF format using the export dropdown button in the period selector toolbar. You can choose the required format using the export dropdown button in stock-chart.

The rendered stock chart can be printed directly using print button in period selector toolbar.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair } from '@syncfusion/ej2-
charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
    },
    primaryXAxis: { majorGridLines: { color: 'transparent' } },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
    title: 'AAPL Stock Price'
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="./index.css" rel="stylesheet">
    <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Disable Export and print

To empty the value of `exportType` for to disable the Export button.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair } from '@syncfusion/ej2-
charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
    },
    primaryXAxis: { majorGridLines: { color: 'transparent' } },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
    title: 'AAPL Stock Price',

```

```
exportType: []
});
stockChart.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Appearance in EJ2 JavaScript Stock chart control

Stock Chart Title

Stock Chart can be given a title using [title](#) property, to show the information about the data plotted.

INDEX.TS

```
import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair } from
 '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
```

```

StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
    },
    primaryXAxis: { majorGridLines: { color: 'transparent' } },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
    title: 'AAPL Stock Price',
    titleStyle: {
        fontFamily: "Arial",
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: "#E27F2D",
        size: '20px'
    }
});
stockChart.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

<!-- markdownlint-disable MD036 -->

Title Customization

The `textStyle` property of stock chart title provides options to customize the `size`, `color`, `fontFamily`, `fontWeight`, `fontStyle`, `opacity`, `textAlignment` and `textOverflow`.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair, IStockChartEventArgs,
ChartTheme } from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
    },
    primaryXAxis: { majorGridLines: { color: 'transparent' } },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
    title: 'AAPL Stock Price',
    titleStyle: {
        fontFamily: 'Arial',
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: '#E27F2D',
        size: '20px',
        textOverflow: 'Wrap'
    },
});
stockChart.appendTo("#element");

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Stock Chart Theme

Changing Stock Chart theme will affect background color, grid lines, tooltip colors and appearance.

[theme](#) property of Stock chart is shipped with several built-in themes such as **Material**, **Fabric**, **Bootstrap**, **HighContrastLight**, **MaterialDark**, **FabricDark**, **FabricDark**, **HighContrast** and **BootstrapDark**.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Crosshair } from '@syncfusion/ej2-
charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);

```

```

StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Crosshair,
Export);
import { chartData } from './datasource.ts';
let stockChart: StockChart = new StockChart({
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent', width: 0 },
    },
    primaryXAxis: { majorGridLines: { color: 'transparent' } },
    series: [
        {
            dataSource: chartData,
            type: 'Candle'
        },
    ],
    title: 'AAPL Stock Price',
    theme: 'HighContrast'
});
stockChart.appendTo("#element");

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="node_modules/@syncfusion/ej2-
inputs/styles/highcontrast.css">
  <link href="node_modules/@syncfusion/ej2-
calendars/styles/highcontrast.css" rel="stylesheet">
  <link href="node_modules/@syncfusion/ej2-
splitbuttons/styles/highcontrast.css" rel="stylesheet">
  <link href="node_modules/@syncfusion/ej2-base/styles/highcontrast.css"
rel="stylesheet">
  <link href="node_modules/@syncfusion/ej2-
navigations/styles/highcontrast.css" rel="stylesheet">
  <link href="node_modules/@syncfusion/ej2-popups/styles/highcontrast.css"
rel="stylesheet">
  <link href="node_modules/@syncfusion/ej2-
buttons/styles/highcontrast.css" rel="stylesheet">
  <link href="index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Axis Customization](#)

<!-- markdownlint-disable MD036 -->

Stock events in EJ2 JavaScript Stock chart control

Stock Events visualizes stock events in stock chart. 'SplineSeries' is used to represent selected data value. You can customize the specific data value using `stockEvents` event.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
IStockChartEventArgs, ChartTheme } from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
    chartArea: { border: { width: 0 } },
    primaryXAxis: { valueType: 'DateTime', majorGridLines: { color:
'transparent' }, crosshairTooltip: { enable: true } },
    primaryYAxis: {
        lineStyle: { color: 'transparent' },
        majorTickLines: { color: 'transparent' },
        crosshairTooltip: { enable: true }
    },
    series: [

```

```

        {
            dataSource: chartData, xName: 'date', yName: 'high', type:
'Spline'
        }
    ],
    stockEvents: [
        { date: new Date(2012, 3, 1), text: 'Q2', description: '2012
Quarter2 starts', type: 'Flag', background: '#6c6d6d', border: { color:
'#6c6d6d' } },
        { date: new Date(2012, 3, 20), text: 'Open', description:
'Markets opened', textStyle: { color: 'white' },
            background: '#f48a21', border: { color: '#f48a21' } },
        { date: new Date(2012, 6, 1), text: 'Q3', description: '2013
Quarter3 starts', type: 'Flag',
            textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' } },
        { date: new Date(2012, 9, 1), text: 'Q4', description: '2013
Quarter4 starts', type: 'Flag',
            textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' } },
        { date: new Date(2012, 7, 30), text: 'G', description: 'Google
stocks bought',
            textStyle: { color: 'white' }, background: '#f48a21', border:
{ color: '#f48a21' } },
        { date: new Date(2012, 10, 1), text: 'Y', description: 'Yahoo
stocks sold', type: 'Square',
            textStyle: { color: 'white' }, background: '#841391', border:
{ color: '#841391' } },
        { date: new Date(2012, 12, 0), text: 'Y2', description: 'Year
2013', type: 'Pin', showOnSeries: false,
            textStyle: { color: 'white' }, background: '#6322e0', border:
{ color: '#6322e0' } },
        { date: new Date(2013, 3, 1), text: 'Q2', description: '2013
Quarter2 starts', type: 'Flag',
            textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' } },
        { date: new Date(2013, 3, 20), text: 'Q2', description: 'Surge
in Stocks', type: 'ArrowUp',
            textStyle: { color: 'white' }, background: '#3ab0f9', border:
{ color: '#3ab0f9' } },
        { date: new Date(2013, 6, 1), text: 'Q3', description: '2013
Quarter3 starts', type: 'Flag',
            textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' } },
        { date: new Date(2013, 9, 1), text: 'Q4', description: '2013
Quarter4 starts', type: 'Flag',
            textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' } },
        { date: new Date(2013, 12, 0), text: 'Y3', description: 'Year
2014', type: 'Pin', showOnSeries: false,
            textStyle: { color: 'white' }, background: '#6322e0', border:
{ color: '#6322e0' } },
        { date: new Date(2014, 3, 1), text: 'Q2', description: '2014
Quarter2 starts', type: 'ArrowDown',
            textStyle: { color: 'white' }, background: '#3ab0f9', border:
{ color: '#3ab0f9' } },
    ]
}

```

```

        { date: new Date(2014, 6, 1), text: 'Q3', description: '2014
Quarter3 starts',
          textStyle: { color: 'white' }, background: '#f48a21', border:
{ color: '#f48a21' } },
        { date: new Date(2014, 9, 1), text: 'Q4', description: '2014
Quarter4 starts', type: 'Flag',
          textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' } },
        { date: new Date(2014, 12, 0), text: 'Y4', description: 'Year
2015', type: 'Pin', showOnSeries: false,
          textStyle: { color: 'white' }, background: '#6322e0', border:
{ color: '#6322e0' } },
        { date: new Date(2014, 2, 2), text: 'End', description: 'Markets
closed', type: 'ArrowDown',
          textStyle: { color: 'white' }, background: '#3ab0f9', border:
{ color: '#3ab0f9' } },
        { date: new Date('2015-01-07'), text: 'A', description: 'This is
event description',
          textStyle: { color: 'white' }, background: '#f48a21', border:
{ color: '#f48a21' } },
        { date: new Date(2015, 1, 2), text: 'Q1', description: 'Add
longer text',
          textStyle: { color: 'white' }, background: '#dd3c9f', border:
{ color: '#dd3c9f' }, type: 'Text' },
        { date: new Date(2015, 2, 12), text: 'Close', description:
'Markets closed',
          textStyle: { color: 'white' }, background: '#f48a21', border:
{ color: '#f48a21' } }
    ],
    seriesType : [],
    indicatorType : [],
    trendlineType: [],
    title: 'AAPL Stock Price',
    crosshair: { enable: true },
  });
  stockChart.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

  <div id="container">
    <div id="element"></div>
  </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stock Events for individual series

By default, stock events will be showed for all series. Now, you can set the stock events for particular series using `seriesIndexes` property.

INDEX.TS

```

import { StockChart } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
import { DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries } from '@syncfusion/ej2-charts';
import { AccumulationDistributionIndicator, AtrIndicator, BollingerBands,
EmaIndicator, MomentumIndicator } from '@syncfusion/ej2-charts';
import { MacdIndicator, RsiIndicator, Trendlines, SmaIndicator,
StochasticIndicator, Export } from '@syncfusion/ej2-charts';
import { TmaIndicator, RangeTooltip, Tooltip, Crosshair,
IStockChartEventArgs, ChartTheme } from '@syncfusion/ej2-charts';
StockChart.Inject(DateTime, AreaSeries, CandleSeries, HiloOpenCloseSeries,
HiloSeries, LineSeries, SplineSeries);
StockChart.Inject(AccumulationDistributionIndicator, AtrIndicator,
BollingerBands, EmaIndicator, MomentumIndicator);
StockChart.Inject(MacdIndicator, RsiIndicator, SmaIndicator,
StochasticIndicator);
StockChart.Inject(Trendlines, TmaIndicator, RangeTooltip, Tooltip,
Crosshair, Export);
let stockChart: StockChart = new StockChart({
  chartArea: { border: { width: 0 } },
  primaryXAxis: { valueType: 'DateTime', majorGridLines: { color:
'transparent' }, crosshairTooltip: { enable: true } },
  primaryYAxis: {
    lineStyle: { color: 'transparent' },
    majorTickLines: { color: 'transparent' },
    crosshairTooltip: { enable: true }
  },
  series: [
    {
      dataSource: chartData, xName: 'date', yName: 'high', type:
'Spline'
    },
    {
      dataSource: chartData, xName: 'date', yName: 'low', type:
'Spline'
    }
  ]
});

```

```

    },
    {
        dataSource: chartData, xName: 'date', yName: 'open', type:
'Spline'
    }
],
stockEvents: [
    { date: new Date(2012, 3, 1), text: 'Q2', description: '2012
Quarter2 starts', type: 'Flag', background: '#6c6d6d', border: { color:
'#6c6d6d' }, seriesIndexes: [0] },
    { date: new Date(2012, 3, 20), text: 'Open', description:
'Markets opened', textStyle: { color: 'white' },
        background: '#f48a21', border: { color: '#f48a21' },
seriesIndexes: [0] },
    { date: new Date(2012, 6, 1), text: 'Q3', description: '2013
Quarter3 starts', type: 'Flag',
        textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' }, seriesIndexes: [0] },
    { date: new Date(2012, 9, 1), text: 'Q4', description: '2013
Quarter4 starts', type: 'Flag',
        textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' }, seriesIndexes: [0] },
    { date: new Date(2012, 7, 30), text: 'G', description: 'Google
stocks bought',
        textStyle: { color: 'white' }, background: '#f48a21', border:
{ color: '#f48a21' }, seriesIndexes: [0] },
    { date: new Date(2012, 10, 1), text: 'Y', description: 'Yahoo
stocks sold', type: 'Square',
        textStyle: { color: 'white' }, background: '#841391', border:
{ color: '#841391' }, seriesIndexes: [0] },
    { date: new Date(2012, 12, 0), text: 'Y2', description: 'Year
2013', type: 'Pin', showOnSeries: false,
        textStyle: { color: 'white' }, background: '#6322e0', border:
{ color: '#6322e0' }, seriesIndexes: [0] },
    { date: new Date(2013, 3, 1), text: 'Q2', description: '2013
Quarter2 starts', type: 'Flag',
        textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' }, seriesIndexes: [0] },
    { date: new Date(2013, 3, 20), text: 'Q2', description: 'Surge
in Stocks', type: 'ArrowUp',
        textStyle: { color: 'white' }, background: '#3ab0f9', border:
{ color: '#3ab0f9' }, seriesIndexes: [1] },
    { date: new Date(2013, 6, 1), text: 'Q3', description: '2013
Quarter3 starts', type: 'Flag',
        textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' }, seriesIndexes: [1] },
    { date: new Date(2013, 9, 1), text: 'Q4', description: '2013
Quarter4 starts', type: 'Flag',
        textStyle: { color: 'white' }, background: '#6c6d6d', border:
{ color: '#6c6d6d' }, seriesIndexes: [1] },
    { date: new Date(2013, 12, 0), text: 'Y3', description: 'Year
2014', type: 'Pin', showOnSeries: false,
        textStyle: { color: 'white' }, background: '#6322e0', border:
{ color: '#6322e0' }, seriesIndexes: [1] },
    { date: new Date(2014, 3, 1), text: 'Q2', description: '2014
Quarter2 starts', type: 'ArrowDown',

```

```

        textStyle: { color: 'white' }, background: '#3ab0f9', border:
    { color: '#3ab0f9' }, seriesIndexes: [1] },
    { date: new Date(2014, 6, 1), text: 'Q3', description: '2014
Quarter3 starts',
        textStyle: { color: 'white' }, background: '#f48a21', border:
    { color: '#f48a21' }, seriesIndexes: [2] },
    { date: new Date(2014, 9, 1), text: 'Q4', description: '2014
Quarter4 starts', type: 'Flag',
        textStyle: { color: 'white' }, background: '#6c6d6d', border:
    { color: '#6c6d6d' }, seriesIndexes: [2] },
    { date: new Date(2014, 12, 0), text: 'Y4', description: 'Year
2015', type: 'Pin', showOnSeries: false,
        textStyle: { color: 'white' }, background: '#6322e0', border:
    { color: '#6322e0' }, seriesIndexes: [2] },
    { date: new Date(2014, 2, 2), text: 'End', description: 'Markets
closed', type: 'ArrowDown',
        textStyle: { color: 'white' }, background: '#3ab0f9', border:
    { color: '#3ab0f9' }, seriesIndexes: [2] },
    { date: new Date('2015-01-07'), text: 'A', description: 'This is
event description',
        textStyle: { color: 'white' }, background: '#f48a21', border:
    { color: '#f48a21' }, seriesIndexes: [2] },
    { date: new Date(2015, 1, 2), text: 'Q1', description: 'Add
longer text',
        textStyle: { color: 'white' }, background: '#dd3c9f', border:
    { color: '#dd3c9f' }, type: 'Text', seriesIndexes: [2] },
    { date: new Date(2015, 2, 12), text: 'Close', description:
'Markets closed',
        textStyle: { color: 'white' }, background: '#f48a21', border:
    { color: '#f48a21' }, seriesIndexes: [2] }
    ],
    seriesType : [],
    indicatorType : [],
    trendlineType: [],
    title: 'AAPL Stock Price',
    crosshair: { enable: true },
});
stockChart.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="./index.css" rel="stylesheet">
  <link href="http://cdn.syncfusion.com/ej2/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="es5-datasource.js" type="text/javascript"></script>
```



```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Series Types](#)

Accessibility in EJ2 JavaScript Stock chart control

The Stock chart control followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Stock chart control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

```
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the control meet the requirement.</div>
<div> - Some features of the control do not meet the requirement.</div>
<div> - The control does not meet the requirement.</div>
```

WAI-ARIA attributes

The Stock chart control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Stock chart control:

- `img` (role)
- `button` (role)
- `region` (role)
- `aria-label` (attribute)
- `aria-hidden` (attribute)
- `aria-pressed` (attribute)

Keyboard interaction

The Stock chart control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Stock chart control.

| **Press** | **To do this** |

| --- | --- |

| **Alt + J** | Moves the focus to the Stock chart element. |

| **Tab** | Moves the focus to the next element in the Stock chart. |

| **Shift + Tab** | Moves the focus to the previous element in the Stock chart. |

| **Down Arrow** | Moves the focus to the data point left side from the selected point. |

| **Up Arrow** | Moves the focus to the data point right side from the selected point. |

| **ESC** | Cancel the tooltip for the data point. |

| **Ctrl + P** | Prints the Stock chart. |

Ensuring accessibility

The Stock chart control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Stock chart control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Stock chart control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

Switch

Accessibility in EJ2 JavaScript Switch control

The Switch component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Switch component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

```

}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>

```

WAI-ARIA attributes

The Switch component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Switch component:

| Attributes | Purpose |
|---------------|--|
| role | Indicates the switch component. |
| aria-disabled | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Switch component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Switch component.

| Press | To do this |
|-------|--|
| Space | When the switch has focus, pressing the Space key changes the state of the switch. |

Ensuring accessibility

The Switch component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Switch component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Switch component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

How To

Change size in EJ2 JavaScript Switch control

The different Switch sizes available are default and small. To reduce the size of default Switch to small, set the [cssClass](#) property to `e-small`.

INDEX.TS

```

import { Switch } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';

```

```
enableRipple(true);
//Small Switch.
let switchObj: Switch = new Switch({ cssClass: 'e-small' });
switchObj.appendTo('#switch1');
//Default Switch.
switchObj = new Switch({});
switchObj.appendTo('#switch2');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Switch</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <table class="size">
      <tbody><tr>
        <td class="lSize">Small</td>
        <td>
          <input type="checkbox" id="switch1">
        </td>
      </tr>
      <tr>
        <td class="lSize">Default</td>
        <td>
          <input type="checkbox" id="switch2">
        </td>
      </tr>
    </tbody></table>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
  margin-left: 10px;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  top: 45%;
  left: 45%;
}
.e-switch-wrapper {
  margin-top: 18px;
}
.size tr td {
  padding: 10px;
}
.size .lSize {
  padding-top: 24px;
  font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
  font-size: 13px;
}
.size .lSize {
  user-select: none;
}
```

Customize the appearance of a switch in EJ2 JavaScript Switch control

You can customize the appearance of the Switch component using the CSS rules. Define your own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

Customize Switch bar and handle

Switch bar and handle can be customized as per requirement using CSS rules. Switch bar and handle customized using `cssClass` property. In the following sample, the `border-radius` CSS property for the Switch bar (`e-switch-inner`) and handle (`e-switch-handle`) elements was changed border radius circle to square shape.

INDEX.TS

```
import { Switch } from '@syncfusion/ej2-buttons';
// Initialize Switch with square cssClass.
let switchObj: Switch = new Switch({ cssClass: 'square' });
switchObj.appendTo('#switch1');
// Initialize Switch with custom-switch cssClass.
switchObj = new Switch({ cssClass: 'custom-switch', checked: true });
switchObj.appendTo('#switch2');
// Initialize Switch with handle-text cssClass.
switchObj = new Switch({ cssClass: 'handle-text' });
switchObj.appendTo('#switch3');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Switch</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/fabric.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="container switch-control">
      <div>
        <h3>Customizing Shape</h3>
      </div>
      <div>
        <label for="switch1" style="padding: 10px 85px 10px 7px">
Square Switch </label>
        <input type="checkbox" id="switch1">
      </div>
      <div>
        <label for="switch2" style="padding: 10px 76px 10px 7px">
Bar and handle </label>
        <input type="checkbox" id="switch2">
      </div>
      <div>
        <label for="switch3" style="padding: 10px 96px 10px 7px">
Handle text </label>
        <input type="checkbox" id="switch3">
      </div>
    </div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
  margin-left: 10px;
}

```

```

}
.switch-control div {
  display: flex;
  align-items: center;
}
.switch-control h3 {
  font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
  text-indent: 23px;
}
.switch-control {
  margin: 90px auto;
  width: 240px;
}
.switch-control label {
  -moz-user-select: none;
  cursor: pointer;
  font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
  font-size: 13px;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  top: 45%;
  left: 45%;
}
/* Square Switch */
.e-switch-wrapper.square .e-switch-inner,
.e-switch-wrapper.square .e-switch-handle {
  border-radius: 0;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.custom-switch {
  width: 50px;
  height: 24px;
}
.e-switch-wrapper.custom-switch .e-switch-handle {
  width: 20px;
  height: 16px;
}
.e-switch-wrapper.custom-switch .e-switch-inner,
.e-switch-wrapper.custom-switch .e-switch-handle {
  border-radius: 0;
}
.e-switch-wrapper.custom-switch .e-switch-handle.e-switch-active {
  left: 42px;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.handle-text {
  width: 58px;
  height: 24px;
}
.e-switch-wrapper.handle-text .e-switch-handle {
  width: 26px;

```



```
height: 20px;
left: 2px;
background-color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-handle {
border-radius: 0;
}
.e-switch-wrapper.handle-text .e-switch-handle.e-switch-active {
left: 46px;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active,
.e-switch-wrapper.handle-text:hover .e-switch-inner.e-switch-active .e-switch-on {
background-color: #4d841d;
border-color: #4d841d;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-off {
background-color: #e3165b;
border-color: #e3165b;
}
.e-switch-wrapper.handle-text .e-switch-inner:after,
.e-switch-wrapper.handle-text .e-switch-inner:before {
font-size: 10px;
position: absolute;
line-height: 21px;
font-family: "Helvetica", sans-serif;
z-index: 1;
height: 100%;
transition: all 200ms cubic-bezier(0.445, 0.05, 0.55, 0.95);
}
.e-switch-wrapper.handle-text .e-switch-inner:before {
content: "OFF";
color: #e3165b;
left: 3px;
}
.e-switch-wrapper.handle-text .e-switch-inner:after {
content: "ON";
right: 5px;
color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:before {
color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:after {
color: #4d841d;
}
.e-switch-wrapper.handle-text:not(.e-switch-disabled):hover .e-switch-handle:not(.e-switch-active) {
background-color: #fff;
}
```

Color the Switch

Switch colors can be customized as per the requirement using CSS rules. Switch bar and handle colors customized using `cssClass` property. In the following sample, the Switch bar (`e-switch-inner`) element background and border colors were changed from default colors.

INDEX.TS

```
import { Switch } from '@syncfusion/ej2-buttons';
// Initialize Switch with bar-color cssClass.
let switchObj: Switch = new Switch({ cssClass: 'bar-color' });
switchObj.appendTo('#switch1');
// Initialize Switch with handle-color cssClass.
switchObj = new Switch({ cssClass: 'handle-color', checked: true });
switchObj.appendTo('#switch2');
// Initialize Switch with custom-iOS cssClass.
switchObj = new Switch({ cssClass: 'custom-iOS', checked: true });
switchObj.appendTo('#switch3');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Switch</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/bootstrap.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="container switch-control">
      <div>
        <h3>Customizing Color</h3>
      </div>
      <div>
        <label for="switch1" style="padding: 10px 85px 10px 0">
Custom color </label>
        <input type="checkbox" id="switch1">
      </div>
      <div>
        <label for="switch2" style="padding: 10px 90px 10px 0">
Handle color </label>
        <input type="checkbox" id="switch2">
      </div>
    </div>
  </div>
```

```

        <label for="switch3" style="padding: 10px 107px 10px 0"> iOS
Switch </label>
        <input type="checkbox" id="switch3">
    </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
    margin-left: 10px;
}
.switch-control div {
    display: flex;
    align-items: center;
}
.switch-control h3 {
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
    text-indent: 23px;
}
.switch-control {
    margin: 90px auto;
    width: 240px;
}
.switch-control label {
    -moz-user-select: none;
    user-select: none;
    cursor: pointer;
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
    font-size: 13px;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
/* Custom color Switch */
.e-switch-wrapper.bar-color .e-switch-inner.e-switch-active,
.e-switch-wrapper.bar-color:hover .e-switch-inner.e-switch-active .e-switch-on {
    background-color: #4d841d;
    border-color: #4d841d;
}

```

```

}
.e-switch-wrapper.bar-color .e-switch-inner,
.e-switch-wrapper.bar-color .e-switch-off {
  background-color: #e3165b;
  border-color: #e3165b;
}
.e-switch-wrapper.bar-color .e-switch-handle {
  background-color: #fff;
}
/* handle color Switch */
.e-switch-wrapper.handle-color .e-switch-handle {
  background-color: #e3165b;
}
.e-switch-wrapper.handle-color .e-switch-handle.e-switch-active {
  background-color: #4d841d
}
.e-switch-wrapper.handle-color .e-switch-inner.e-switch-active,
.e-switch-wrapper.handle-color:hover .e-switch-inner.e-switch-active .e-switch-on {
  background-color: #fff;
  border-color: #ccc;
}
.e-switch-wrapper.handle-color .e-switch-inner,
.e-switch-wrapper.handle-color .e-switch-off {
  background-color: #fff;
  border-color: #ccc;
}
/* iOS Switch */
.e-switch-wrapper.custom-iOS .e-switch-inner.e-switch-active,
.e-switch-wrapper.custom-iOS:hover .e-switch-inner.e-switch-active .e-switch-on {
  background-color: #3df865;
  border-color: #3df665;
}
.e-switch-wrapper.custom-iOS {
  width: 42px;
  height: 24px;
}
.e-switch-wrapper.custom-iOS .e-switch-handle {
  width: 20px;
  height: 20px;
}
.e-switch-wrapper.custom-iOS .e-switch-handle.e-switch-active {
  margin-left: -22px;
}

```

Enable ripple for switch label in EJ2 JavaScript Switch control

By default, label with ripple effect is not available in Switch. You can achieve this using `rippleMouseHandler`

method.

The following example illustrates how to enable ripple effect for labels in Switch component.

INDEX.TS

```

import { Switch, rippleMouseHandler } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Switch component.
let switchObj: Switch = new Switch({ checked: true });
// Render initialized Switch.
switchObj.appendTo('#switch1');
// Function to handle ripple effect for Switch with label.
document.querySelector('.lSize label').addEventListener('mouseup',
rippleHandler);
document.querySelector('.lSize label').addEventListener('mousedown',
rippleHandler);
function rippleHandler(e: MouseEvent): void {
    let rippleSpan: Element =
this.parentElement.nextElementSibling.querySelector('.e-ripple-container');
    if (rippleSpan) {
        rippleMouseHandler(e, rippleSpan);
    }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Switch</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript UI Controls">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="styles.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <table class="size">
            <tbody><tr>
                <td class="lSize"><label for="switch1">USB
Tethering</label></td>
                <td>
                    <input type="checkbox" id="switch1">
                </td>
            </tr>
        </tbody></table>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {

```

```

        ele.style.visibility = "visible";
    }
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
    margin-left: 10px;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
.e-switch-wrapper {
    margin-top: 18px;
}
.size tr td {
    padding: 10px;
}
.size .lSize {
    padding-top: 24px;
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
    font-size: 13px;
}
.size .lSize label {
    cursor: pointer;
    user-select: none;
}

```

Enable rtl in EJ2 JavaScript Switch control

Switch component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in Switch component.

INDEX.TS

```

import { Switch } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Switch component.
let switchObj: Switch = new Switch({ enableRtl: true });
// Render initialized Switch.
switchObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Switch</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="styles.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <input type="checkbox" id="element">
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
    margin-left: 10px;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
.e-switch-wrapper {
    margin-top: 18px;
}

```

Set disabled state in EJ2 JavaScript Switch control

Switch can be disabled by setting the [disabled](#) property to true.

The following example illustrates how to disable support in Switch component.

INDEX.TS

```
import { Switch } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Switch component with disabled support.
let switchObj: Switch = new Switch({ disabled: true });
// Render initialized Switch.
switchObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Switch</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input type="checkbox" id="element">
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
  margin-left: 10px;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
  top: 45%;
  left: 45%;
}
```



```
.e-switch-wrapper {
  margin-top: 18px;
}
```

Submit name and value in form in EJ2 JavaScript Switch control

The [name](#) attribute of the Switch is used to group Switches. When the Switches are grouped in form, the checked items [value](#) attribute will post to the server on form submit. The disabled and unchecked Switch values will not be sent to the server on form submit.

In the following code snippet, USB and Wi-Fi in the [checked](#) state, and Bluetooth is in disabled state. Values that are in checked state only be sent on form submit.

INDEX.TS

```
import { Switch, Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Name and Value attribute in form submit.
let switchObj: Switch = new Switch({ name: 'Tethering', value: 'USB',
checked: true });
switchObj.appendTo('#switch1');
switchObj = new Switch({ name: 'Hotspot', value: 'Wi-Fi', checked: true });
switchObj.appendTo('#switch2');
switchObj = new Switch({ name: 'Tethering', value: 'Bluetooth', disabled:
true });
switchObj.appendTo('#switch3');
let button: Button = new Button({ isPrimary: true });
button.appendTo('#btnElement');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Switch</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <form>
      <table class="size">
        <tbody><tr>
```

```

        <td class="lSize">USB</td>
        <td>
            <input type="checkbox" id="switch1">
        </td>
    </tr>
    <tr>
        <td class="lSize">Wi-Fi</td>
        <td>
            <input type="checkbox" id="switch2">
        </td>
    </tr>
    <tr>
        <td class="lSize">Bluetooth</td>
        <td>
            <input type="checkbox" id="switch3">
        </td>
    </tr>
    <tr>
        <td>
            <button id="btnElement">Submit</button>
        </td>
    </tr>
</tbody></table>
</form>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
button {
    margin: 20px 0 0 5px;
}
.size tr td {
    padding: 10px;
}
.size .lSize {
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
}

```

```
font-size: 13px;
}
.size .lSize label {
  user-select: none;
}
```

Change switch state using toggle method in EJ2 JavaScript Switch control

This section explains about how to toggle between the switch states using [toggle](#) method.

INDEX.TS

```
import { Switch } from '@syncfusion/ej2-buttons';
//Set text in switch toggle states.
let switchObj: Switch = new Switch({ onLabel: 'ON', offLabel: 'OFF' });
switchObj.appendTo('#switch1');
switchObj.toggle();
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Switch</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/fabric.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/bootstrap.css" rel="stylesheet">
  <link href="styles.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input type="checkbox" id="switch1">
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
```

```
}  
#loader {  
  color: #008cff;  
  height: 40px;  
  width: 30%;  
  position: absolute;  
  top: 45%;  
  left: 45%;  
}  
.e-switch-wrapper {  
  margin-top: 18px;  
}
```

Switch triggers [change](#) event on every state stage to perform custom operations.

Tabs

Getting started in EJ2 JavaScript Tab control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization

The Essential JS 2 JavaScript components can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder `myapp` for Essential JS 2 JavaScript components.

Step 2: You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

Syntax:

Script: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js`

Styles: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css`

Example:

Script: `C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-navigations/dist/global/ej2-navigations.min.js`

Styles: `C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-navigations/styles/material.css`

Step 3: Create a folder `myapp/resources` and copy/paste the global scripts and styles from the above installed location to `myapp/resources` location.

Step 4: Create a HTML page (`index.html`) in `myapp` location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Tab's global script -->
<script src="resources/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the `Tab` element and initiate the `Essential JS 2 Tab` component in the `index.html` by using following code

Initialize the Tab using JSON items collection

The Tab can be rendered by defining a JSON array. The item is rendered with [header](#) text and [content](#) for each Tab using [items](#) property.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Tab's global script -->
<script src="resources/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
```

```
<!-- Add the HTML <div> element -->
<div id="element"></div>
<script>
var tabObj = new ej.navigations.Tab({
items: [
{
header: { 'text': 'Twitter' },
content: 'Twitter is an online social networking service that enables users to send and read short 140-
character ' +
'messages called "tweets". Registered users can read and post tweets, but those who are unregistered
can only read ' +
'them. Users access Twitter through the website interface, SMS or mobile device app Twitter Inc. is
based in San ' +
'Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack
Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained
worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets a day in 2012.The service also handled 1.6
billion ' +
'search queries per day.'
},
{
header: { 'text': 'Facebook' },
content: 'Facebook is an online social networking service headquartered in Menlo Park, California. Its
website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow
students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders had initially limited the
websites' +
'membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League,
and Stanford ' +
'University. It gradually added support for students at various other universities and later to high-school
students.'
},
{
header: { 'text': 'WhatsApp' },
```

content: 'WhatsApp Messenger is a proprietary cross-platform instant messaging client for smartphones that operates ' +

'under a subscription business model. It uses the Internet to send text messages, images, video, user location and ' +

'audio media messages to other users using standard cellular mobile numbers. As of February 2016, WhatsApp had a user ' +

'base of up to one billion,[10] making it the most globally popular messaging application. WhatsApp Inc., based in ' +

'Mountain View, California, was acquired by Facebook Inc. on February 19, 2014, for approximately US\$19.3 billion.'

}

]

});

//Render initialized Tab component

tabObj.appendTo('#element');

</script>

</body>

</html>

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Tab** component.

Using [CDN link for script and style reference](#)

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

Script: <http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css>

Step 3: Create a HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `Tab` element and initiate the `Essential JS 2 Tab` component in the `index.html` by using following code.

INDEX.HTML

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>

    <title>Essential JS 2</title>
    <!-- Essential JS 2 material theme -->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet" type="text/css"/>
    <!-- Essential JS 2 Tab's global script -->
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
  </head>
  <body>
    <!-- Add the HTML <div> element -->
    <div id="element"></div>
    <script>
      var tabObj = new ej.navigations.Tab({
        items: [
          {
            header: { 'text': 'Twitter' },
            content: 'Twitter is an online social networking
service that enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read
and post tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website
interface, SMS or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the
world. Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched
in July 2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million
tweets a day in 2012. The service also handled 1.6 billion ' +
'search queries per day.'
          },
          {
            header: { 'text': 'Facebook' },
            content: 'Facebook is an online social networking
service headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with
his Harvard College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes. The founders had initially limited the website's ' +
'membership to Harvard students, but later expanded it
to colleges in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students
at various other universities and later to high-school students.'
          },
        ],
      });
    </script>
  </body>
</html>

```



```

        {
            header: { 'text': 'WhatsApp' },
            content: 'WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates ' +
                'under a subscription business model. It uses the
Internet to send text messages, images, video, user location and ' +
                'audio media messages to other users using standard
cellular mobile numbers. As of February 2016, WhatsApp had a user ' +
                'base of up to one billion, [10] making it the most
globally popular messaging application. WhatsApp Inc., based in ' +
                'Mountain View, California, was acquired by Facebook
Inc. on February 19, 2014, for approximately US* 9.3 billion.'
        }
    }
});
//Render initialized Tab component
tabObj.appendTo('#element');
</script>
</body>
</html>

```

Step 4: Now, run the `index.html` in web browser, it will render the `Essential JS 2 Tab` component.

Initialize the Tab using HTML elements

The Tab component can be rendered based on the given HTML element using `id` as `target`.

Header section must be enclosed with in a wrapper element using `e-tab-header` class and corresponding content must be mapped with `e-content` class.

You need to follow the below structure of HTML elements to render the Tab,

,

`<div id='tabhtmlmarkup'>` --> Root Tab element

`<div class="e-tab-header">` --> Tab header

`<div>` --> Header Item

`</div>`

`</div>`

`<div class="e-content">` --> Tab content

`<div>` --> Content Item

`</div>`

`</div>`

`</div>`

,

- Add the HTML template data with its id attribute and add it in your `index.html` file to initialize the Tab.

INDEX.HTML

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>

    <title>Essential JS 2</title>
    <!-- Essential JS 2 material theme -->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet" type="text/css"/>
    <!-- Essential JS 2 Tab's global script -->
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
  </head>
  <body>
    <!-- Add the HTML <div> element -->
    <div id="element">
      <div class="e-tab-header">
        <div> Twitter </div>
        <div> Facebook </div>
        <div> WhatsApp </div>
      </div>
      <div class="e-content">
        <div>
          Twitter is an online social networking service that
enables users to send and read short 140-character messages called 'tweets'.
Registered users can read and post tweets, but those who are unregistered
can only read them. Users access Twitter through the website interface, SMS
or mobile device app Twitter Inc. is based in San Francisco and has more
than 25 offices around the world. Twitter was created in March 2006 by Jack
Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006.
The service rapidly gained worldwide popularity, with more than 100 million
users posting 340 million tweets a day in 2012.The service also handled 1.6
billion search queries per day.
        </div>
        <div>
          Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was launched on
February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and
fellow students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.The founders had initially limited the website's membership to
Harvard students, but later expanded it to colleges in the Boston area, the
Ivy League, and Stanford University. It gradually added support for students
at various other universities and later to high-school students.
        </div>
        <div>
          WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates under a subscription
business model. It uses the Internet to send text messages, images, video,

```

```

user location and audio media messages to other users using standard
cellular mobile numbers. As of February 2016, WhatsApp had a user base of up
to one billion,[10] making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
by Facebook Inc. on February 19, 2014, for approximately US$ 9.3 billion.
    </div>
</div>
</div>
<script>
    var tabObj = new ej.navigations.Tab();
    //Render initialized Tab component
    tabObj.appendTo('#element');
</script>
</body>
</html>

```

Adaptive in EJ2 JavaScript Tab control

The following section explains about rendering Tab when its width exceeds the viewable area or particularly in a given [width](#). The available modes are as follows:

- Scrollable
- Popup

Scrollable

The default overflow mode is Scrollable. Scrollable display mode supports displaying the Tab header items in a single line with horizontal scrolling enabled, when the item overflows to the available space.

- The right and left navigation arrow is added at the start and end of the Tab header through which user can navigate towards overflowed items of the Tab header.
- You can also see the overflowed items using touch and swipe action on the header and content section.
- By default, navigation icon in the left direction is disabled, you can see the overflowed items by moving in the right direction.
- By clicking the arrow or by holding the arrow continuously, you can see the overflowed items.



- In devices the navigation icons are not available. You can touch and swipe to see the overflowed items of the Tab header.

HTML

C SHARP(C#)

JAVA

VB.NET

XAMARIN

ASP.NET

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    heightAdjustMode: 'Auto',
    overflowMode: 'Scrollable',
    items: [
        {
            header: { 'text': 'HTML' },
            content: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web ' +
'pages. Along with CSS, and JavaScript, HTML is a cornerstone
technology, used by most websites to create visually ' +
'engaging web pages, user interfaces for web applications, and
user interfaces for many mobile applications.[1] Web ' +
'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
'website semantically along with cues for presentation, making
it a markup language, rather than a programming language.'
        },
        {
            header: { 'text': 'C Sharp(C#)' },
            content: 'C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its development ' +
'team is led by Anders Hejlsberg. The most recent version is C#
5.0, which was released on August 15, 2012.'
        },
        {
            header: { 'text': 'Java' },
            content: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle ' +
'Corporation, that provides a system for developing application
software and deploying it in a cross-platform computing ' +
'environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
'provide many features of native applications and can be
embedded in HTML pages.'
        },
        {
            header: { 'text': 'VB.Net' },
            content: 'The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also ' +
```

```

        'includes a command-line VB.NET compiler. The most recent
version is VB 2012, which was released on August 15, 2012.'
    },
    {
        header: { 'text': 'Xamarin' },
        content: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers that ' +
        'created Mono,[4] Mono for Android and MonoTouch that are cross-
platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android, iOS,
and Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
        'as of May 2015.'
    },
    {
        header: { 'text': 'ASP.NET' },
        content: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\'s Active Server Pages (ASP) technology. ASP.NET
is built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.'
    },
    {
        header: { 'text': 'ASP.NET MVC' },
        content: 'The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC,
ASP.NET Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project
is called ASP.NET vNext.'
    },
    {
        header: { 'text': 'JavaScript' },
        content: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as part of ' +
        'web browsers so that client-side scripts could interact with
the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop
applications.'
    }
}
]);
tabObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Tab">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
    </div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Popup

The Popup is the another type of [overflowMode](#) in which the Tab container holds the items that can be placed within the available space. The rest of the overflowing items for which there is no space to fit within the viewing area are moved to overflow popup container.

- The items placed in popup can be viewed by opening the popup with the help of drop-down icon given at the end of the Tab header.
- If the popup height exceeds the height of the visible area, you can scroll through the popup items and select one.

HTML

C SHARP(C#)

JAVA

VB.NET

XAMARIN



HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
  heightAdjustMode: 'Auto',
  overflowMode: 'Popup',
  items: [
    {
      header: { 'text': 'HTML' },
      content: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web ' +
      'pages. Along with CSS, and JavaScript, HTML is a cornerstone
technology, used by most websites to create visually ' +
      'engaging web pages, user interfaces for web applications, and
user interfaces for many mobile applications.[1] Web ' +
      'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
      'website semantically along with cues for presentation, making
it a markup language, rather than a programming language.'
    },
    {
      header: { 'text': 'C Sharp(C#)' },
      content: 'C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its development ' +
      'team is led by Anders Hejlsberg. The most recent version is C#
5.0, which was released on August 15, 2012.'
    },
    {
      header: { 'text': 'Java' },
      content: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle ' +
      'Corporation, that provides a system for developing application
software and deploying it in a cross-platform computing ' +
      'environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
      'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
      'provide many features of native applications and can be
embedded in HTML pages.'
    },
    {
      header: { 'text': 'VB.Net' },
      content: 'The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also ' +
```

```

        'includes a command-line VB.NET compiler. The most recent
version is VB 2012, which was released on August 15, 2012.'
    },
    {
        header: { 'text': 'Xamarin' },
        content: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers that ' +
        'created Mono,[4] Mono for Android and MonoTouch that are cross-
platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android, iOS,
and Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
        'as of May 2015.'
    },
    {
        header: { 'text': 'ASP.NET' },
        content: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\'s Active Server Pages (ASP) technology. ASP.NET
is built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.'
    },
    {
        header: { 'text': 'ASP.NET MVC' },
        content: 'The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC,
ASP.NET Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The project
is called ASP.NET vNext.'
    },
    {
        header: { 'text': 'JavaScript' },
        content: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as part of ' +
        'web browsers so that client-side scripts could interact with
the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop
applications.'
    }
}
]);
tabObj.appendTo('#element');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Tab">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element">
    </div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to prevent content swipe selection](#)
- [Collapsible Tab](#)

Header in EJ2 JavaScript Tab control

This section explains about modifying the style of Tab header, and configuring its icons and positions.

Styles

You can customize header styles by adding predefined classes in the Tab root element. The pre-defined CSS class names are as follows:

- **e-fill:** The Selected Tab header background is set as solid fill.
- **e-background:** Tab header has a solid fill background, and the selected header has a highlighted border.

- **e-background e-accent:** Tab header has a solid fill background, and the selected header has a highlighted border with accent color.

If the above custom style classes are not included in the root element, the default style is applied to the Tab items.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-
dropdowns';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let tabObj: Tab = new Tab({
  items: [
    {
      header: { 'text': 'Twitter' },
      content: 'Twitter is an online social networking service
that enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and
post tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface,
SMS or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets
a day in 2012. The service also handled 1.6 billion ' +
'search queries per day.'
    },
    {
      header: { 'text': 'Facebook' },
      content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes. The founders had initially limited the website's ' +
'membership to Harvard students, but later expanded it to
colleges in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students at
various other universities and later to high-school students.'
    },
    {
      header: { 'text': 'WhatsApp' },
      content: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates ' +
'under a subscription business model. It uses the Internet
to send text messages, images, video, user location and ' +
'audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user ' +
'base of up to one billion, [10] making it the most globally
popular messaging application. WhatsApp Inc., based in ' +
'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.'
```

```

    }
  ]
});
tabObj.appendTo('#element');
let headerStyles: DropDownList = new DropDownList({
  width: '90%',
  change: ()=>{ changeHeaderStyle();}
});
headerStyles.appendTo('#headerStyles');
function changeHeaderStyle(e: ChangeEventArgs): void {
  removeStyleClass();
  if (headerStyles.value === 'fill') {
    tabObj.element.classList.add('e-fill');
  } else if (headerStyles.value === 'background') {
    tabObj.element.classList.add('e-background');
  } else if (headerStyles.value === 'accent') {
    tabObj.element.classList.add('e-background');
    tabObj.element.classList.add('e-accent');
  }
}
function removeStyleClass(): void {
  tabObj.element.classList.remove('e-fill');
  tabObj.element.classList.remove('e-background');
  tabObj.element.classList.remove('e-accent');
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="default">
      <div class="row">
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
          <label> Header Style </label>
        </div>
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">

```

```

        <select id="headerStyles">
            <option value="default">Default</option>
            <option value="fill">Fill</option>
            <option value="background">Background</option>
            <option value="accent">Accent</option>
        </select>
    </div>
</div>
</div>
<br>
<div id="element"></div>
<div id="result"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Icon positions

You can customize the position of the Tab header icons using the [iconPosition](#) property. This property depends on the header items [iconCss](#) property. By default, Tab header icon is placed on left position. The position values are as follows:

- **Left:** Icon is placed on the left of the Tab header item.
- **Right:** Icon is placed on the right of the Tab header item.
- **Top:** Icon is placed on the top of the Tab header item.
- **Bottom:** Icon is placed on the bottom of the Tab header item.

INDEX.TS

```

import { Tab } from '@syncfusion/ej2-navigations';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-
dropdowns';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let tabObj: Tab = new Tab({
    items: [
        {
            header: { 'text': 'Twitter', 'iconCss': 'e-twitter' },
            content: 'Twitter is an online social networking service
that enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and
post tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface,
SMS or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, ' +

```

```

        'with more than 100 million users posting 340 million tweets
a day in 2012.The service also handled 1.6 billion ' +
        'search queries per day.'
    },
    {
        header: { 'text': 'Facebook', 'iconCss': 'e-facebook' },
        content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
        'launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo ' +
        'Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes.The founders had initially limited the website\'\'s ' +
        'membership to Harvard students, but later expanded it to
colleges in the Boston area, the Ivy League, and Stanford ' +
        'University. It gradually added support for students at
various other universities and later to high-school students.'
    },
    {
        header: { 'text': 'WhatsApp', 'iconCss': 'e-whatsapp' },
        content: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates ' +
        'under a subscription business model. It uses the Internet
to send text messages, images, video, user location and ' +
        'audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user ' +
        'base of up to one billion,[10] making it the most globally
popular messaging application. WhatsApp Inc., based in ' +
        'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.'
    }
]
});
//Render initialized Tab component
tabObj.appendTo('#element');
let iconPosition: DropDownList = new DropDownList({
    width: '90%',
    change: ()=>{ changeHeaderStyle();}
});
iconPosition.appendTo('#iconPosition');
function changeHeaderStyle(e: ChangeEventArgs): void {
    let items: any = tabObj.items;
    for(let i: number = 0; i < items.length; i++) {
        tabObj.items[i].header.iconPosition = iconPosition.value;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Tab</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Typescript Toolbar Controls">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="default">
            <div class="row">
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <label>Icon Position</label>
                </div>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <select id="iconPosition">
                        <option value="left">Left</option>
                        <option value="right">Right</option>
                        <option value="top">Top</option>
                        <option value="bottom">Bottom</option>
                    </select>
                </div>
            </div>
            <br>
            <div id="element"></div>
            <div id="result"></div>
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to customize selected tab styles](#)

Localization in EJ2 JavaScript Tab control

Localization library allows to localize the default text content of the Tab to different cultures using the [locale](#) property. In Tab, the close button's tooltip text alone will be localize based on culture. The close button is shown on tab header when enabled [showCloseButton](#) property.

| Locale key | en-US (default) |

|-----|-----|

| closeButtonTitle | Close |

Loading translations

To load translation object in an application use `load` function of `L10n` class.

In the below sample, `French` culture is set to Tab and change the close button's tooltip text.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-
dropdowns';
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let tabObj: Tab = new Tab({
  items: [
    {
      header: { 'text': 'Twitter' },
      content: 'Twitter is an online social networking service
that enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and
post tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface,
SMS or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in
July 2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets
a day in 2012. The service also handled 1.6 billion ' +
'search queries per day.'
    },
    {
      header: { 'text': 'Facebook' },
      content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris
Hughes. The founders had initially limited the website's ' +
'membership to Harvard students, but later expanded it to
colleges in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students at
various other universities and later to high-school students.'
    },
    {
      header: { 'text': 'WhatsApp' },
      content: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates ' +
'under a subscription business model. It uses the Internet
to send text messages, images, video, user location and ' +
'audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user ' +
'base of up to one billion, [10] making it the most globally
popular messaging application. WhatsApp Inc., based in ' +
```

```

        'Mountain View, California, was acquired by Facebook Inc. on
        February 19, 2014, for approximately US$19.3 billion.'
    }
    ],
    locale: 'fr-BE',
    showCloseButton: true
});
// Load French culture for tab close button tooltip text
L10n.load({
    'fr-BE': {
        'tab': {'closeButtonText': "Fermer"}
    }
});
L10n.load({
    'ar-AR': {
        'tab': {'closeButtonText': "إغلاق زر العنوان"}
    }
});
L10n.load({
    'de-DE': {
        'tab': {'closeButtonText': "Schließen Schaltfläche Titel"}
    }
});
//Render initialized Tab component
tabObj.appendTo('#element');
let localeData: { [key: string]: Object }[] = [
    { Id: 'ar-AR', Game: 'Arabic' },
    { Id: 'fr-BE', Game: 'French' },
    { Id: 'de-DE', Game: 'German' }
];
// initialize DropDownList component
let dropDownListObject: DropDownList = new DropDownList({
    width: '250px',
    //set the data to dataSource property
    dataSource: localeData,
    //set height to popup list
    fields: { text: 'Game', value: 'Id' },
    popupHeight: '200px',
    //set width to popup list
    popupWidth: '250px',
    change: change,
    value: 'fr-BE',
    // set placeholder to DropDownList input element
    placeholder: "Select a Language"
});
function change(e: ChangeEventArgs): void {
    if(e.value == "ar-AR")
        tabObj.locale = "ar-AR"
    if (e.value == "fr-BE")
        tabObj.locale = "fr-BE"
    if (e.value == "de-DE")
        tabObj.locale = "de-DE"
}
// render initialized DropDownList
dropDownListObject.appendTo('#ddlelement');

```


INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <input type="text" tabindex="1" id="ddlelement">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Orientation in EJ2 JavaScript Tab control

This section explains about modifying the position and modes of Tab header.

It allows placing the header section inside the Tab component at different positions by using the [headerPlacement](#) property. The available positions are as follows:

- **Top:** Tab header items can be arranged horizontally, and their content can be placed after the header.
- **Bottom:** Tab header items can be arranged horizontally, and their content can be placed before the header.
- **Left:** Tab header items can be arranged vertically, and their content can be placed after the header.
- **Right:** Tab header items can be arranged vertically, and their content can be placed before the header.

It is also adaptable to the available space when the tab items exceed the view space. You can customize the modes by using `overflowMode` property. The available modes are as follows:

- Scrollable
- Popup

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let tabObj: Tab = new Tab({
    heightAdjustMode: 'None',
    height: 150,
    width: 700,
    items: [
        {
            header: { 'text': 'HTML' },
            content: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web ' +
'pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
'engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web ' +
'browsers can read HTML files and render them into visible
or audible web pages. HTML describes the structure of a ' +
'website semantically along with cues for presentation,
making it a markup language, rather than a programming language.'
        },
        {
            header: { 'text': 'C Sharp(C#)' },
            content: 'C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its development ' +
'team is led by Anders Hejlsberg. The most recent version is
C# 5.0, which was released on August 15, 2012.'
        },
        {
            header: { 'text': 'Java' },
            content: 'Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by Oracle ' +
'Corporation, that provides a system for developing
application software and deploying it in a cross-platform computing ' +
'environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
'enterprise servers and supercomputers. While less common,
Java applets run in secure, sandboxed environments to ' +
'provide many features of native applications and can be
embedded in HTML pages.'
        },
        {
            header: { 'text': 'VB.Net' },
            content: 'The command-line compiler, VBC.EXE, is installed
as part of the freeware .NET Framework SDK. Mono also ' +
'includes a command-line VB.NET compiler. The most recent
version is VB 2012, which was released on August 15, 2012.'
        },
        {
            header: { 'text': 'Xamarin' },
```

```

        content: 'Xamarin is a San Francisco, California based
software company created in May 2011[3] by the engineers that ' +
        'created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications
(often called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android,
iOS, and Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1
million developers in more than 120 countries around the World ' +
        'as of May 2015.'
    },
    {
        header: { 'text': 'ASP.NET' },
        content: 'ASP.NET is an open-source server-side web
application framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002
with version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'\'s Active Server Pages (ASP) technology.
ASP.NET is built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.'
    },
    {
        header: { 'text': 'ASP.NET MVC' },
        content: 'The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC,
ASP.NET Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The
project is called ASP.NET vNext.'
    },
    {
        header: { 'text': 'JavaScript' },
        content: 'JavaScript (JS) is an interpreted computer
programming language. It was originally implemented as part of ' +
        'web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop
applications.'
    }
]
});
tabObj.appendTo('#element');
let headerPosition: DropDownList = new DropDownList({
    change: (e: ChangeEventArgs)=> {
        tabObj.headerPlacement = (<{ [key: string]: any;
}>e.itemData).text;
        tabObj.dataBind();
    }
})

```

```

    });
    headerPosition.appendTo('#headerPosition');
    let mode: DropDownList = new DropDownList({
        change: (arg: ChangeEventArgs) => {
            tabObj.overflowMode = (<{ [key: string]: any;
        }>arg.itemData).text;
            tabObj.dataBind();
        }
    });
    mode.appendTo('#Mode');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Tab Controls">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" type="text/css">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="default" style="padding-top:20px">
      <div class="row" style="float:left">
        <label> Header Position </label>
        <div>
          <select id="headerPosition">
            <option value="top">Top</option>
            <option value="bottom">Bottom</option>
            <option value="left">Left</option>
            <option value="right">Right</option>
          </select>
        </div>
      </div>
      <div class="row" style="float:right">
        <label> Mode </label>
        <div>
          <select id="Mode">
            <option value="scrollable">Scrollable</option>
            <option value="popup">Popup</option>
          </select>
        </div>
      </div>
    </div>
  </div>

```

```

        </div>
        <div id="element"></div>
        <br>
        <br>
    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop in EJ2 JavaScript Tab control

The Tab component allows you to drag and drop any item by setting [allowDragAndDrop](#) to **true**. Items can be reordered to any place by dragging and dropping them onto the desired location.

- If you need to prevent dragging action for a particular item, the [onDragStart](#) event can be used which will trigger when the item drag is started. If you need to prevent dropping action for a particular item, the [dragged](#) event can be used which will trigger when the drag action is stopped.
- The [dragArea](#) defines the area in which the draggable element movement will be occurring. Outside that area will be restricted for the draggable element movement.
- The [onDragStart](#) event will be triggered before dragging the item from Tab.
- The [dragging](#) event will be triggered when the Tab item is being dragged.
- The [dragged](#) event will be triggered when the Tab item is dropped on the target element successfully.

In the following sample, the [allowDragAndDrop](#) property is enabled.

INDEX.TS

```

import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    heightAdjustMode: 'Auto',
    allowDragAndDrop: true,
    dragArea: '#container',
    items: [
        {
            header: { text: 'India' },
            content: 'India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.'
        },
        {

```

```

        header: { text: 'Australia' },
        content: 'Australia, officially the Commonwealth of Australia,
is a country comprising the mainland of the Australian continent, the island
of Tasmania and numerous smaller islands. It is the world sixth-largest
country by total area. Neighboring countries include Indonesia, East Timor
and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New
Caledonia to the north-east; and New Zealand to the south-east.'
    },
    {
        header: { text: 'USA' },
        content: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.'
    },
    {
        header: { text: 'France' },
        content: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.'
    }
],
});
tabObj.appendTo('#draggableTab');

```

INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
    <title>Essential JS 2 for Tab </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Tab UI Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

    <div id="container">
        <div id="draggableTab"></div>
    </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop item between tabs

It is possible to drag and drop the tab items between two tabs, by manually saving those dropped items as new tab item data through the `addTab` method of Tab and removing the dragged item through the `removeTab` method of Tab.

In this example, we have used the tab control as an external source, and the item from the tab component is dragged and dropped onto another Tab. Therefore, it is necessary to use the `onDragStart` and `dragged` event of the Tab component, where we can form an event object and save it using the `addTab` method of the Tab and remove the dragged item through `removeTab` method of Tab using the dragged item index.

INDEX.TS

```

import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { Tab, DragEventArgs } from '@syncfusion/ej2-navigations';
let firstTabObj: Tab = new Tab({
    heightAdjustMode: 'Auto',
    allowDragAndDrop: true,
    dragArea: '#container',
    onDragStart: firstTabdragStart,
    dragged: firstTabDragStop,
    items: [
        {
            header: { text: 'India' },
            content: 'India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.'
        },
        {
            header: { text: 'Australia' },
            content: 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the north-east; and New Zealand to the south-east.'
        },
        {
            header: { text: 'USA' },

```

```

        content: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.'
    },
    {
        header: { text: 'France' },
        content: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.'
    }
],
});
firstTabObj.appendTo('#firstTab');
let secondTabObj: Tab = new Tab({
    heightAdjustMode: 'Auto',
    dragArea: '#container',
    allowDragAndDrop: true,
    onDragStart: secondTabDragStart,
    dragged: secondTabDragStop,
    items: [
        {
            header: { text: 'HTML' },
            content: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web pages. Along with
CSS, and JavaScript, HTML is a cornerstone technology, used by most websites
to create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages. HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language.'
        },
        {
            header: { text: 'C Sharp(C#)' },
            content: 'C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its development team is led
by Anders Hejlsberg. The most recent version is C# 5.0, which was released
on August 15, 2012.'
        },
        {
            header: { text: 'Java' },
            content: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross-platform computing environment. Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers. While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages.'
        }
    ]
});

```



```

    },
    {
        header: { text: 'VB.Net' },
        content: 'The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also includes a command-line
VB.NET compiler. The most recent version is VB 2012, which was released on
August 15, 2012.'
    }
],
});
secondTabObj.appendTo('#secondTab');
let firstTabitem: object[];
let secondTabitem: object[];
let dragItemIndex: number;
let dragItemContainer: HTMLElement;
function firstTabdragStart(args: DragEventArgs) {
    firstTabitem = [firstTabObj.items[args.index]];
    args.draggedItem.style.visibility = 'hidden';
    dragItemContainer = <HTMLElement>args.draggedItem.closest('.e-tab');
}
function firstTabDragStop(args: DragEventArgs) {
    if (!isNullOrUndefined(args.target.closest('.e-tab')) &&
!dragItemContainer.isSameNode(args.target.closest('.e-tab'))) {
        args.cancel = true;
        let TabElement: HTMLElement = <HTMLElement>args.target.closest('.e-
tab');
        let dropItem: HTMLElement = <HTMLElement>args.target.closest('.e-
toolbar-item');
        if (TabElement != null && dropItem != null) {
            dragItemIndex =
Array.prototype.indexOf.call(firstTabObj.element.querySelectorAll('.e-
toolbar-item'), args.draggedItem);
            let dropItemContainer: Element = args.target.closest('.e-
toolbar-items');
            let dropItemIndex: number = (dropItemContainer != null) ?
(Array.prototype.slice.call(dropItemContainer.querySelectorAll('.e-toolbar-
item')).indexOf(dropItem) : '');
            secondTabObj.addTab(firstTabitem, dropItemIndex);
            firstTabObj.removeTab(dragItemIndex);
        }
    }
}
function secondTabDragStart(args: DragEventArgs) {
    secondTabitem = [secondTabObj.items[args.index]];
    args.draggedItem.style.visibility = 'hidden';
    dragItemContainer = <HTMLElement>args.draggedItem.closest('.e-tab');
}
function secondTabDragStop(args: DragEventArgs) {
    if (!isNullOrUndefined(args.target.closest('.e-tab')) &&
!dragItemContainer.isSameNode(args.target.closest('.e-tab'))) {
        args.cancel = true;
        let TabElement: HTMLElement = <HTMLElement>args.target.closest('.e-
tab');
        let dropItem: HTMLElement = <HTMLElement>args.target.closest('.e-
toolbar-item');
        if (TabElement != null && dropItem != null) {

```

```

        dragItemIndex =
Array.prototype.indexOf.call(secondTabObj.element.querySelectorAll('.e-
toolbar-item'), args.draggedItem);
        let dropItemContainer: Element = args.target.closest('.e-
toolbar-items');
        let dropItemIndex: number = (dropItemContainer != null) ?
(Array.prototype.slice.call(dropItemContainer.querySelectorAll('.e-toolbar-
item'))).indexOf(dropItem) : '';
        firstTabObj.addTab(secondTabitem, dropItemIndex);
        secondTabObj.removeTab(dragItemIndex);
    }
}
}

```

INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
    <title>Essential JS 2 for Tab </title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 for Tab UI Control">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

    <div id="container" style="height: auto">
        <div id="firstTab" style="height:auto">
        </div>
        <div id="secondTab" style="height:auto">
        </div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop items to external source

It is possible to drag and drop the items to any of the external sources from the Tab, by manually saving those dropped items as new node data through the `addNodes` method of Treeview and removing the dragged item through the `removeTab` method of Tab.

In this example, we have used the tree view control as an external source, and the item from the tab component is dragged and dropped onto the child nodes of the tree view component. Therefore, it is necessary to use the `dragged` event of the Tab component, where we can form an event object and save it using the `addNodes` method of the Treeview and remove the dragged item through the `removeTab` method of Tab using the dragged item index.

INDEX.TS

```
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { Tab, DragEventArgs, TabItemModel } from '@syncfusion/ej2-
navigations';
import { TreeView } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    created: onTabCreate,
    heightAdjustMode: 'Auto',
    dragArea: '#container',
    allowDragAndDrop: true,
    dragged: tabDragStop,
    items: [
        {
            header: { 'text': 'India' },
            content: 'India officially the Republic of India, is a country
in South Asia. It is the seventh-largest country by area, the second-most
populous country with over 1.2 billion people, and the most populous
democracy in the world. Bounded by the Indian Ocean on the south, the
Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it
shares land borders with Pakistan to the west; China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.'
        },
        {
            header: { 'text': 'Australia' },
            content: 'Australia, officially the Commonwealth of Australia,
is a country comprising the mainland of the Australian continent, the island
of Tasmania and numerous smaller islands. It is the world sixth-largest
country by total area. Neighboring countries include Indonesia, East Timor
and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New
Caledonia to the north-east; and New Zealand to the south-east.'
        },
        {
            header: { 'text': 'USA' },
            content: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.'
        },
        {
            header: { 'text': 'France' },
            content: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
```

```

and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.'
    }
    ],
    });
    tabObj.appendTo('#draggableTab');
    let data: { [key: string]: Object }[] = [
        { text: 'Hennessey Venom', id: 'list-01' },
        { text: 'Bugatti Chiron', id: 'list-02' },
        { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
        { text: 'SSC Ultimate Aero', id: 'list-04' },
        { text: 'Koenigsegg CCR', id: 'list-05' },
        { text: 'McLaren F1', id: 'list-06' },
        { text: 'Aston Martin One- 77', id: 'list-07' },
        { text: 'Jaguar XJ220', id: 'list-08' },
        { text: 'McLaren P1', id: 'list-09' },
        { text: 'Ferrari LaFerrari', id: 'list-10' },
    ];
    let treeViewInstance: TreeView = new TreeView({
        fields: { dataSource: data, id: 'id', text: 'text' },
        cssClass: 'treeview-external-drop-tab'
    });
    treeViewInstance.appendTo('#draggableTreeview');
    let i: number = 0;
    function onTabCreate(): void {
        let tabElement: HTMLElement = document.getElementById('draggableTab');
        if (!isNullOrUndefined(tabElement)) {
            tabElement.querySelector('.e-tab-header').classList.add('e-
droppable');
            tabElement.querySelector('.e-content').classList.add('tab-content');
        }
    }
    function tabDragStop(args: DragEventArgs) {
        let dragTabIndex: number =
Array.prototype.indexOf.call(tabObj.element.querySelectorAll('.e-toolbar-
item'), args.draggedItem);
        let dragItem: TabItemModel = tabObj.items[dragTabIndex];
        let dropNode: HTMLElement =
<HTMLElement>args.target.closest('#draggableTreeview .e-list-item');
        if (dropNode != null && !args.target.closest('#draggableTab .e-toolbar-
item')) {
            args.cancel = true;
            let dropContainer: NodeListOf<Element> =
(document.querySelector('.treeview-external-drop-
tab')).querySelectorAll('.e-list-item');
            let dropIndex: number = Array.prototype.indexOf.call(dropContainer,
dropNode);
            let newNode: { [key: string]: Object }[] = [{ id: 'list' + i, text:
dragItem.header.text }];
            tabObj.removeTab(dragTabIndex);
            treeViewInstance.addNodes(newNode, 'Treeview' , dropIndex);
        }
    }
}

```

INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
  <title>Essential JS 2 for Tab </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Tab UI Control">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link href="index.css" rel="stylesheet">

  <div id="container" style="height: auto">
    <div id="draggableTab"></div>
    <div id="draggableTreeview"></div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Drag and drop items from external source

It is possible to drag and drop the items from any of the external sources into the Tab, by manually saving those dropped items as new item data through the `addTab` method of Tab and removing the dragged node through the `removeNodes` method of Treeview.

In this example, we have used the tree view control as an external source, and the child nodes from the tree view component are dragged and dropped onto the Tab. Therefore, it is necessary to use the `nodeDragStop` event of the Treeview component, where we can form an event object and save it using the `addTab` method of the Tab and remove the dragged node through the `removeNodes` method of Treeview.

INDEX.TS

```

import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { Tab, TabItemModel } from '@syncfusion/ej2-navigations';
import { TreeView, DragAndDropEventArgs } from '@syncfusion/ej2-
navigations';
let tabObj: Tab = new Tab({
  heightAdjustMode: 'Auto',
  dragArea: '#container',
  items: [

```

```

        {
            header: { 'text': 'India' },
            content: 'India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.'
        },
        {
            header: { 'text': 'Australia' },
            content: 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the north-east; and New Zealand to the south-east.'
        },
        {
            header: { 'text': 'USA' },
            content: 'The United States of America (USA or U.S.A.), commonly called the United States (US or U.S.) and America, is a federal republic consisting of fifty states and a federal district. The 48 contiguous states and the federal district of Washington, D.C. are in central North America between Canada and Mexico. The state of Alaska is west of Canada and east of Russia across the Bering Strait, and the state of Hawaii is in the mid-North Pacific. The country also has five populated and nine unpopulated territories in the Pacific and the Caribbean.'
        },
        {
            header: { 'text': 'France' },
            content: 'France, officially the French Republic is a sovereign state comprising territory in western Europe and several overseas regions and territories. The European part of France, called Metropolitan France, extends from the Mediterranean Sea to the English Channel and the North Sea, and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo metres and as of August 2015 has a population of 67 million, counting all the overseas departments and territories.'
        }
    ],
    });
    tabObj.appendTo('#draggableTab');
    let data: { [key: string]: Object }[] = [
        { text: 'Hennessey Venom', id: 'list-01' },
        { text: 'Bugatti Chiron', id: 'list-02' },
        { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
        { text: 'SSC Ultimate Aero', id: 'list-04' },
        { text: 'Koenigsegg CCR', id: 'list-05' },
        { text: 'McLaren F1', id: 'list-06' },
        { text: 'Aston Martin One- 77', id: 'list-07' },
        { text: 'Jaguar XJ220', id: 'list-08' },
        { text: 'McLaren P1', id: 'list-09' },
        { text: 'Ferrari LaFerrari', id: 'list-10' },
    ]

```

```

];
let treeViewObj: TreeView = new TreeView({
  fields: { dataSource: data, id: 'id', text: 'text' },
  allowDragAndDrop: true,
  dragArea: '#container',
  nodeDragStop: onNodeDragStop,
  nodeDragging: onNodeDrag,
  cssClass: 'treeview-external-drop-tab'
});
treeViewObj.appendTo('#draggableTreeview');
function onNodeDragStop(args: DragAndDropEventArgs): void {
  let dropElement: HTMLElement =
<HTMLElement>args.target.closest('#draggableTab .e-toolbar-item');
  if (dropElement != null) {
    let tabElement: HTMLElement =
document.querySelector('#draggableTab');
    let dropItemIndex: number =
[[]].slice.call(tabElement.querySelectorAll('.e-toolbar-
item')).indexOf(dropElement);
    let newTabItem: TabItemModel[] = [{
      header: { 'text': args.draggedNodeData.text.toString() },
      content: args.draggedNodeData.text.toString() + ' Content'
    }];
    tabObj.addTab(newTabItem, dropItemIndex);
    treeViewObj.removeNodes([args.draggedNode]);
    args.cancel = true;
  } else {
    let dropNode: HTMLElement =
<HTMLElement>args.target.closest('#draggableTreeview .e-list-item ');
    if (!isNullOrUndefined(dropNode) && args.dropIndicator === 'e-drop-
in') {
      args.cancel = true;
    }
  }
}
function onNodeDrag(args: DragAndDropEventArgs): void {
  if (!isNullOrUndefined(args.target.closest('.tab-content'))) {
    args.dropIndicator = 'e-no-drop';
  } else if (!isNullOrUndefined(args.target.closest('#draggableTab .e-tab-
header'))) {
    args.dropIndicator = 'e-drop-in';
  }
}
}

```

INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
  <title>Essential JS 2 for Tab </title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 for Tab UI Control">

```

```
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link href="index.css" rel="stylesheet">

<div id="container" style="height: auto">
  <div id="draggableTab"></div>
  <div id="draggableTreeview"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Accessibility in EJ2 JavaScript Tab control

The Tab control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Tab control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |


```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The
control does not meet the requirement.</div>
```

ARIA attributes

Tab control is designed by considering [WAI-ARIA](#) standard. Tab is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices. The following list of attributes are added in the Tab.

| Roles and Attributes | Functionalities |

| --- | --- |

| tablist | Attribute is set to the Tab header element that describes actual role of the element. |

| tab | Attribute is set to the Tab items element to indicates an interactive element inside a **tablist** that, when activated, displays its associated **tabpanel**. |

| tabpanel | Attribute is set to the Tab content that describes the role for viewing the active content. |

| aria-orientation | Attribute is set to the Tab header element indicates the Tab header orientation. Default value of this attribute is horizontal. |

| aria-selected | Attribute set to the Tab items to indicates the selection state for Tab items. Active Tab is set to true for this attribute. |

| aria-labelledby | Attribute is set to the Tab content element to indicates the associated Tab header for the content. |

| aria-controls | Attribute is set to the Tab items element to indicates the associated **tabpanel** for the header. |

| aria-haspopup | Attribute is set to the Popup element to indicates the popup mode in the Tab. The default value of this attribute is false. If popup mode is enabled, the attribute value is set to true. |

| aria-disabled | Attribute set to the Tab items to It indicates the disabled state of the Tab. |

Keyboard interaction

By default, keyboard navigation is enabled. This control implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Tab element, you can use the following key combination for interacting with the Tab.

Key	Description
Left	Moves focus to the previous Tab. If focus is on the first Tab, the focus will not move to any Tab.
Right	Moves focus to the next Tab. If focus is on the last Tab element, the focus will not move to any Tab.
Enter or Space	Selects the Tab if it is not selected. Opens the popup dropdown icon if it is focussed. Select the Tab item as active when popup item is focussed.
Esc(Escape)	Closes the popup if popup is in opened state.
Down or Up	When the popup is open and focused, it will move to previous/next Tab items of the popup in the vertical direction.
Home	Moves focus to the first Tab.
End	Moves focus to the last Tab.
Shift + F10	If popup mode is enabled, it opens the popup when the Tab is focused.
Delete	Deletes the Tab, if close button is enabled in Tab header.
Tab	To Move focus through the interactive elements.
Shift + Tab	To Move focus through the interactive elements.

Ensuring accessibility

The Tab control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Tab control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Tab control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

Style in EJ2 JavaScript Tab control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing Tab

Use the following CSS to customize the Tab.

```
`css
.e-tab {
border: 5px solid rgb(173, 255, 47);
}
```

Customizing the Tab items

Use the following CSS to customize the header items of Tab.

```
`css
.e-tab .e-tab-header .e-toolbar-items {
background: #9faed8;
border: 2px solid blue;
}
`
```

Use the following CSS to customize the content items of Tab.

```
`css
.e-tab .e-content .e-item {
color: #a78515;
font-size: 14px;
}
`
```

Customizing Tab's header

Use the following CSS to customize the header of Tab control.

```
`css
.e-tab .e-tab-header {
background: #badfba !important;
}
`
```

Customizing Tab's header icon

Use the following CSS to customize the header item icon of Tab control.

```
`css
.e-tab .e-tab-header .e-toolbar-item .e-tab-icon {
color: #badfba !important;
}
`
```

Customizing Tab's content

Use the following CSS to customize the content of Tab control.

```
`css
.e-tab .e-content {
background: #d1f6d1 !important;
}
```

```
}  
,
```

Customizing the hover state of Tab control

Use the following CSS to customize the tab item when hovering.

```
`css  
.e-tab .e-tab-header .e-toolbar-item .e-tab-wrap:hover {  
background: #d1f6d1 !important;  
}  
,
```

Use the following CSS to customize the tab item popup icon when hovering.

```
`css  
.e-tab .e-tab-header .e-hor-nav .e-popup-up-icon:hover,  
.e-tab .e-tab-header .e-hor-nav .e-popup-down-icon:hover {  
background: #d1f6d1 !important;  
}  
,
```

Customizing selected item of Tab control

Use the following CSS to customize the selected tab item.

```
`css  
.e-tab .e-tab-header .e-toolbar-item.e-active {  
background: #d1f4d1;  
}  
,
```

Use the following CSS to customize the selected tab item text and icon.

```
`css  
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-text,  
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-icon {  
color: green !important;  
}  
,
```

How To

Load content through post in EJ2 JavaScript Tab control

The Tab supports to load external contents through AJAX library. Refer to the following steps.

- Import the Ajax module from ej2-base and initialize with URL path.

- Get the data from Ajax Success event, then initialize the Tab with retrieved external path data.

INDEX.TS

```
import {Tab} from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { Ajax } from '@syncfusion/ej2-base';
enableRipple(true);
let ajax: Ajax = new Ajax('./ajax.html', 'GET', true);
ajax.send().then();
ajax.onSuccess = (data: string): void => {
    let ctn2: string = data;
    let tabObj: Tab = new Tab({
        items: [
            { header: { 'text' : 'Facebook'},
              content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his Harvard
College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The
founders had initially limited the website\'\'s ' +
'membership to Harvard students, but later expanded it to colleges
in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students at various
other universities and later to high-school students.'},
            { header: { 'text' : 'Whatsapp'},
              content: 'WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates ' +
'under a subscription business model. It uses the Internet to send
text messages, images, video, user location and ' +
'audio media messages to other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user ' +
'base of up to one billion,[10] making it the most globally popular
messaging application. WhatsApp Inc., based in ' +
'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.' },
            { header: { 'text' : 'Twitter'}, content: ctn2 },
        ]
    });
    tabObj.appendTo('#element');
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Tab</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Tab">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Prevent content swipe selection in EJ2 JavaScript Tab control

We can prevent the tab selection on touch swipe action by using the Tab [selecting](#) event. Refer the below sample with preventing swipe selection.

INDEX.TS

```

import { Tab, selectEventArgs } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    selecting: select,
    items: [
        {
            header: { 'text': 'Twitter' },
            content: 'Twitter is an online social networking service that
enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and post
tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface, SMS
or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in July
2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets a
day in 2012. The service also handled 1.6 billion ' +
'search queries per day.'
        },
        {
            header: { 'text': 'Facebook' },
            content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The
founders had initially limited the website\'s ' +

```

```

        'membership to Harvard students, but later expanded it to
        colleges in the Boston area, the Ivy League, and Stanford ' +
        'University. It gradually added support for students at various
        other universities and later to high-school students.'
    },
    {
        header: { 'text': 'WhatsApp' },
        content: 'WhatsApp Messenger is a proprietary cross-platform
        instant messaging client for smartphones that operates ' +
        'under a subscription business model. It uses the Internet to
        send text messages, images, video, user location and ' +
        'audio media messages to other users using standard cellular
        mobile numbers. As of February 2016, WhatsApp had a user ' +
        'base of up to one billion,[10] making it the most globally
        popular messaging application. WhatsApp Inc., based in ' +
        'Mountain View, California, was acquired by Facebook Inc. on
        February 19, 2014, for approximately US$19.3 billion.'
    }
]
});
tabObj.appendTo('#element');
function select(e: selectEventArgs): void {
    if (e.isSwiped) {
        e.cancel = true;
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Tab</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Tab">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize selected tab styles in EJ2 JavaScript Tab control

You can customize the Tab style by overriding its header and active tab CSS classes. Define HTML string for adding animation and customizing the Tab header and pass it to [text](#) property. Now you can override the style using custom CSS classes added to the Tab elements.

You can add the custom class into Tab component using [cssClass](#) property which is used to customize the Tab component.

INDEX.TS

```

import {Tab} from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
  cssClass: "e-tab-custom-class",
  items: [
    {
      header: { 'text': '<div><div class="e-image e-andrew"></div><div class="e-title fade-in">Andrew</div></div>' },
      content: 'Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.'
    },
    {
      header: { 'text': '<div><div class="e-image e-margaret"></div><div class="e-title fade-in">Margaret</div></div>' },
      content: 'Margaret holds a BA in English literature from Concordia College (1958) and an MA from the American Institute of Culinary Arts (1966). She was assigned to the London office temporarily from July through November 1992.'
    },
    {
      header: { 'text': '<div><div class="e-image e-janet"></div><div class="e-title fade-in">Janet</div></div>' },
      content: 'Janet has a BS degree in chemistry from Boston College (1984). She has also completed a certificate program in food retailing management. Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992.'
    }
  ]
});
tabObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Tab">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="element"></div>
        <br><br>
        <div id="result"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize tab scroll step in EJ2 JavaScript Tab control

Tab supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize **ScrollStep** property for scrolling distance. Refer to the following code example.

- By using Tab scrollStep property, pass a required value to customize tab scrollStep.

INDEX.TS

```

import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    scrollStep: 50,
    items: [
        {
            header: { 'text': 'HTML' },
            content: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web ' +
'pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually ' +
'engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web ' +
'browsers can read HTML files and render them into visible
or audible web pages. HTML describes the structure of a ' +
'website semantically along with cues for presentation,
making it a markup language, rather than a programming language.'
        },
        {

```

```

        header: { 'text': 'C Sharp(C#)' },
        content: 'C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its development ' +
        'team is led by Anders Hejlsberg. The most recent version is
C# 5.0, which was released on August 15, 2012.'
    },
    {
        header: { 'text': 'Java' },
        content: 'Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by Oracle ' +
        'Corporation, that provides a system for developing
application software and deploying it in a cross-platform computing ' +
        'environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
        'enterprise servers and supercomputers. While less common,
Java applets run in secure, sandboxed environments to ' +
        'provide many features of native applications and can be
embedded in HTML pages.'
    },
    {
        header: { 'text': 'VB.Net' },
        content: 'The command-line compiler, VBC.EXE, is installed
as part of the freeware .NET Framework SDK. Mono also ' +
        'includes a command-line VB.NET compiler. The most recent
version is VB 2012, which was released on August 15, 2012.'
    },
    {
        header: { 'text': 'Xamarin' },
        content: 'Xamarin is a San Francisco, California based
software company created in May 2011[3] by the engineers that ' +
        'created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language ' +
        'Infrastructure (CLI) and Common Language Specifications
(often called Microsoft .NET). With a C#-shared codebase, ' +
        'developers can use Xamarin tools to write native Android,
iOS, and Windows apps with native user interfaces and share ' +
        'code across multiple platforms.[5] Xamarin has over 1
million developers in more than 120 countries around the World ' +
        'as of May 2015.'
    },
    {
        header: { 'text': 'ASP.NET' },
        content: 'ASP.NET is an open-source server-side web
application framework designed for web development to produce ' +
        'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
        'and web services. It was first released in January 2002
with version 1.0 of the .NET Framework, and is the successor ' +
        'to Microsoft\'s Active Server Pages (ASP) technology.
ASP.NET is built on the Common Language Runtime (CLR), allowing ' +
        'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
        'ASP.NET components to process SOAP messages.'
    },
    {
        header: { 'text': 'ASP.NET MVC' },

```

```

        content: 'The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the ' +
        'model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is ' +
        'proprietary. In the later versions of ASP.NET, ASP.NET MVC,
ASP.NET Web API, and ASP.NET Web Pages (a platform using ' +
        'only Razor pages) will merge into a unified MVC 6.The
project is called ASP.NET vNext.'
    },
    {
        header: { 'text': 'JavaScript' },
        content: 'JavaScript (JS) is an interpreted computer
programming language. It was originally implemented as part of ' +
        'web browsers so that client-side scripts could interact
with the user, control the browser, communicate ' +
        'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
        'both game development and the creation of desktop
applications.'
    }
]
});
tabObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Tab Control">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
  <br><br>
  <div id="result"></div>

  <script>
var ele = document.getElementById('container');
if(ele) {

```

```

    ele.style.visibility = "visible";
  }
  </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Create wizard using tab in EJ2 JavaScript Tab control

Tab items can be disabled dynamically by passing the index and boolean value with the [enableTab](#) method and also passing index or HTML element to select an item from the tab using [select](#) method.

In the below Wizard sample, each Tab is integrated with required components to complete the reservation. Each field is provided with validation for all mandatory option to proceed to next tabs. Using Tab item's template property the components are added into content.

```
`ts
```

/ Initializing Tab with header and contents bind to template div for adding other components /

```
tabObj = new Tab({ heightAdjustMode: 'None', height: 390, showCloseButton: false,
```

```
selecting: tabSelected,
```

```
items: [
```

```
{ header: { 'text': 'New Booking' }, content: '#booking' },
```

```
{ header: { 'text': 'Train List' }, content: '#selectTrain', disabled: true },
```

```
{ header: { 'text': 'Add Passenger' }, content: '#details', disabled: true },
```

```
{ header: { 'text': 'Make Payment' }, content: '#confirm', disabled: true }
```

```
]
```

```
});
```

```
tabObj.appendTo('#element');
```

```
,
```

Create the following contents for each tab in the wizard.

1. Search tab:

Created with [DropDownList](#) to select the source, destination and type of ticket. A [DatePicker](#) for choosing the date of journey.

2. Train tab:

Based on the selected start and end point, populated Grid with random list of available seats and train list. Initially define the columns and row selected event for validating, after the source and destination chosen update the [dataSource](#) for the Grid.

3. Passenger tab:

A table with Textbox, Numeric, DropDownList for adding passenger name, age, gender and preferred berth/seat. Add validation on entering passenger details to proceed.

4. Payment tab:

Calculate the ticket cost based on location, passenger count and ticket type. Generate data for Grid with passenger details, train number and ticket cost summary.

You can go back on each tab using buttons available in it and tabs are [disabled](#) to navigate through tab header click actions. Once you end the wizard all the data get cleared and wizard goes back to starting tab.

INDEX.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { Tab, SelectEventArgs } from '@syncfusion/ej2-navigations';
import { DatePicker } from '@syncfusion/ej2-calendars';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { Grid, RowSelectEventArgs } from '@syncfusion/ej2-grids';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { isNullOrUndefined as isNOU } from '@syncfusion/ej2-base';

let tabObj: Tab;
let availTrainGrid: Grid;
let ticketDetailGrid: Grid;
let endPoint: DropDownList;
let journeyDate: DatePicker;
let ticketType: DropDownList;
let startPoint: DropDownList;
let alertDlg: Dialog;
let today : Date = new Date();
let locations: any;
let selectedTrain: any;
let quota: any = [
    { id: '1', text: 'Business Class' },
    { id: '2', text: 'Economy Class' },
    { id: '3', text: 'Common Class' }
];
let gender: any = [
    { id: '1', text: 'Male' },
    { id: '2', text: 'Female' }
];
let berths: any = [
    { id: '1', text: 'Upper' },
    { id: '2', text: 'Lower' },
    { id: '3', text: 'Middle' },
    { id: '4', text: 'Window' },
    { id: '5', text: 'Aisle' }
];
let cities: any = [
    {name: 'Chicago', fare: 300},
    {name: 'San Francisco', fare: 125},
    {name: 'Los Angeles', fare: 175},
    {name: 'Seattle', fare: 250},
    {name: 'Florida', fare: 150}
];

renderComponents();
```

```

function renderComponents(): void {
    /* Initialize Tab with disabled headers for the wizard */
    tabObj = new Tab({ heightAdjustMode: 'None', height: 390,
showCloseButton: false,
    selecting: tabSelected,
    items: [
        { header: { 'text': 'New Booking' }, content: '#booking' },
        { header: { 'text': 'Train List' }, content: '#selectTrain',
disabled: true },
        { header: { 'text': 'Add Passenger' }, content: '#details',
disabled: true },
        { header: { 'text': 'Make Payment' }, content: '#confirm',
disabled: true }
    ]
    });
    tabObj.appendTo('#element');
    /* Initialize the components for creating wizard */
    startPoint = new DropDownList({
        width: '100%', dataSource: cities, floatLabelType: 'Auto',
placeholder: 'From',
        fields: { text: 'name', value: 'name' }
    });
    startPoint.appendTo('#startPoint');
    endPoint = new DropDownList({
        width: '100%', dataSource: cities, floatLabelType: 'Auto',
placeholder: 'To',
        fields: { text: 'name', value: 'name' }
    });
    endPoint.appendTo('#endPoint');
    journeyDate = new DatePicker({
        width: '100%', floatLabelType: 'Auto', placeholder: 'Journey Date',
min: new Date(today.getTime()),
        max: new Date(today.getTime() + 60 * 24 * 60 * 60 * 1000),
        focus: () => { journeyDate.show(); }
    });
    journeyDate.appendTo('#journey_date');
    ticketType = new DropDownList({
        dataSource: quota, placeholder: 'Ticket Type', floatLabelType:
'Auto', fields: { text: 'text', value: 'text' }
    });
    ticketType.appendTo('#ticket_type');
    alertDlg = new Dialog({
        header: 'Success', width: 250, isModal: true, visible: false,
showCloseIcon: true,
        content: 'Your payment successfully processed', target:
document.getElementById('container'), created: dlgCreated
    });
    alertDlg.appendTo('#alertDialog');
    alertDlg.hide();
    availTrainGrid = new Grid({
        width: '100%',
        columns: [
            { field: 'TrainNo', headerText: 'Train No', width: 120, type:
'number' },
            { field: 'Name', width: 140, headerText: 'Name' },
            { field: 'Departure', headerText: 'Departure', width: 120 },
            { field: 'Arrival', headerText: 'Arrival', width: 140 },

```

```

        { field: 'Availability', headerText: 'Availability', width: 140,
type: 'number' }
    ],
    rowSelected: trainSelected
});
availTrainGrid.appendTo('#availableTrain');
let age1: NumericTextBox = new NumericTextBox({ min: 1, max: 100, value:
18, format: 'n0', showSpinButton: false });
age1.appendTo('#pass_age1');
let age2: NumericTextBox = new NumericTextBox({ min: 1, max: 100, value:
18, format: 'n0', showSpinButton: false });
age2.appendTo('#pass_age2');
let age3: NumericTextBox = new NumericTextBox({ min: 1, max: 100, value:
18, format: 'n0', showSpinButton: false });
age3.appendTo('#pass_age3');
let gender1: DropDownList = new DropDownList({
    dataSource: gender, text: 'Male', fields: { text: 'text', value:
'text' }
});
gender1.appendTo('#pass_gender1');
let gender2: DropDownList = new DropDownList({
    dataSource: gender, text: 'Male', fields: { text: 'text', value:
'text' }
});
gender2.appendTo('#pass_gender2');
let gender3: DropDownList = new DropDownList({
    dataSource: gender, text: 'Male', fields: { text: 'text', value:
'text' }
});
gender3.appendTo('#pass_gender3');
let berth1: DropDownList = new DropDownList({
    dataSource: berths, placeholder: 'Optional', fields: { text: 'text',
value: 'text' }
});
berth1.appendTo('#pass_berth1');
let berth2: DropDownList = new DropDownList({
    dataSource: berths, placeholder: 'Optional', fields: { text: 'text',
value: 'text' }
});
berth2.appendTo('#pass_berth2');
let berth3: DropDownList = new DropDownList({
    dataSource: berths, placeholder: 'Optional', fields: { text: 'text',
value: 'text' }
});
berth3.appendTo('#pass_berth3');
ticketDetailGrid = new Grid({
    width: '100%',
    columns: [
        { field: 'TrainNo', headerText: 'Train No', width: 120, type:
'number' },
        { field: 'PassName', width: 140, headerText: 'Name' },
        { field: 'Gender', headerText: 'Gender', width: 120 },
        { field: 'Berth', headerText: 'Berth', width: 140 }
    ],
});
ticketDetailGrid.appendTo('#ticketDetailGrid');

```

```

        document.getElementById('searchNext').onclick = (e: any) => {
            tabNavigations(e); };
        document.getElementById('bookTickets').onclick = (e: any) => {
            tabNavigations(e); };
        document.getElementById('confirmTickets').onclick = (e: any) => {
            tabNavigations(e); };
        document.getElementById('makePayment').onclick = (e: any) => {
            tabNavigations(e); };
        document.getElementById('goToSearch').onclick = (e: any) => {
            tabNavigations(e); };
        document.getElementById('goBackToBook').onclick = (e: any) => {
            tabNavigations(e); };
        document.getElementById('goBackDetails').onclick = (e: any) => {
            tabNavigations(e); };
    }
    function tabSelected(e: SelectEventArgs): void {
        if (e.isSwiped) {
            e.cancel = true;
        }
    }
}
function dlgCreated(): void {
    alertDlg.buttons = [{
        buttonModel: { content: 'Ok', isPrimary: true },
        click: (() => {
            alertDlg.hide();
            tabObj.enableTab(0, true);
            tabObj.enableTab(1, false);
            tabObj.enableTab(2, false);
            tabObj.enableTab(3, false);
            tabObj.select(0);
        })
    }
    ]};
}
function tabNavigations(args: any): void {
    switch (args.target.id) {
        case 'searchNext':
            /* Validate the Source, Destination, Date and Class chosen and
            proceed only if all the fields are selected */
            if (!isNOU(startPoint.value) && !isNOU(endPoint.value) &&
                !isNOU(ticketType.value) && !isNOU(journeyDate.value)) {
                if (!isNOU(startPoint.value) && startPoint.value ===
                endPoint.value) {
                    document.getElementById('err1').innerText = '* Arrival
                    point can\'t be same as Departure';
                } else {
                    tabObj.enableTab(0, false);
                    tabObj.enableTab(1, true);
                    filterTrains(args);
                    tabObj.select(1);
                    document.getElementById('err1').innerText = '';
                    document.getElementById('err2').innerText = '';
                }
            } else {
                document.getElementById('err1').innerText = '* Please fill
                all the details before proceeding';
            }
            break;
    }
}

```



```

        case 'bookTickets':
            /* Based on the selected station generate Grid content to display
            trains available */
            if (availTrainGrid.getSelectedRecords() === undefined ||
            availTrainGrid.getSelectedRecords().length === 0) {
                document.getElementById('err2').innerText = '* Select your
            convenient train';
            } else {
                tabObj.enableTab(2, true);
                tabObj.select(2);
                tabObj.enableTab(1, false);
                document.getElementById('err2').innerText = '';
            }
            break;
        case 'confirmTickets':
            /* Get the Passenger details and validate the fields must not be
            left empty */
            let name: any = document.getElementById('pass_name1');
            let age: any = document.getElementById('pass_age1');
            let gender: any = document.getElementById('pass_gender1');
            if (name.value === '' || age.value === '' || gender.value ===
            '') {
                document.getElementById('err3').innerText = '* Please enter
            passenger details';
            } else {
                tabObj.enableTab(3, true);
                tabObj.select(3);
                tabObj.enableTab(2, false);
                document.getElementById('err3').innerText = '';
                finalizeDetails(args);
            }
            break;
        case 'makePayment':
            alertDlg.show();
            break;
        case 'goToSearch':
            /* Go back to change class, date or boarding places */
            selectedTrain = [];
            tabObj.enableTab(0, true);
            tabObj.select(0);
            tabObj.enableTab(1, false);
            break;
        case 'goBackToBook':
            /* Change the preferred train chosen already */
            tabObj.enableTab(1, true);
            tabObj.select(1);
            tabObj.enableTab(2, false);
            break;
        case 'goBackDetails':
            /* Update passenger detail before confirming the payment */
            tabObj.enableTab(2, true);
            tabObj.select(2);
            tabObj.enableTab(3, false);
            break;
    }
}
function filterTrains(args: any): void {

```

```

/* Generating trains based on source and destination chosen */
let result: Object[] = [];
let fromCity: string = startPoint.text;
let toCity: string = endPoint.text;
let count: number = Math.floor((Math.random() * 3) + 2);
for (let i: number = 0; i < count; i++) {
    let details : any = [];
    details.TrainNo = Math.floor((Math.random() * 20) + 19000);
    details.Name = 'Train ' + i;
    details.Departure = fromCity;
    details.Arrival = toCity;
    details.Availability = Math.floor((Math.random() * 20) + 20);
    result.push(details);
}
availTrainGrid.dataSource = result;
}
function finalizeDetails(args: any): void {
    /* Get the passenger details and update table with name and other
    details for confirmation */
    let reserved: Object[] = [];
    let passCount: any = 0;
    for (let i: number = 1; i <= 3; i++) {
        let name: any = document.getElementById('pass_name' + i);
        let berthSelected: any = document.getElementById('pass_berth' + i);
        let gender: any = document.getElementById('pass_gender' + i);
        if (name.value !== '') {
            let details: any = [];
            let berth: string = berthSelected.value;
            details.TrainNo = selectedTrain.TrainNo.toString();
            details.PassName = name.value;
            details.Gender = gender.value;
            details.Berth = (berth === '') ? 'Any' : berth;
            reserved.push(details);
            passCount++;
        }
        let calcFare: any = 0;
        for (let j: number = 0; j < cities; j++) {
            if (startPoint.value === cities[j].name) {
                calcFare = calcFare + cities[j].fare;
            }
            if (endPoint.value === cities[j].name) {
                calcFare = calcFare + cities[j].fare;
            }
        }
        let displayAmt : any = document.getElementById('amount');
        if (ticketType.value === 'Economy Class') {
            displayAmt.innerText = "Total payable amount: $" + passCount *
(300 + calcFare);
        } else if (ticketType.value === 'Business Class') {
            displayAmt.innerText = "Total payable amount: $" + passCount *
(500 + calcFare);
        } else if (ticketType.value === 'Common Class') {
            displayAmt.innerText = "Total payable amount: $" + passCount *
(150 + calcFare);
        }
    }
    ticketDetailGrid.dataSource = reserved;
}

```

```

}
function trainSelected(args: RowSelectEventArgs): void {
    selectedTrain = args.data;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Tab">
  <meta name="author" content="Syncfusion">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" type="text/css">
  <link href="index.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="alertDialog"></div>
    <div id="element"></div>
  </div>
  <div id="booking" style="display: none">
    <div class="wizard-title">Plan your journey</div>
    <div class="responsive-align">
      <div class="row">
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
          <input id="startPoint" class="e-input" type="text">
        </div>
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
          <input id="endPoint" class="e-input" type="text">
        </div>
      </div>
      <div class="row">
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
          <input id="journey_date" class="e-input" type="text">
        </div>
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
          <input id="ticket_type" class="e-input" type="text">
        </div>
      </div>
    </div>
  </div>

```

```

        <div class="btn-container">
            <button id="searchNext" class="e-btn">Search Train</button>
        </div>
        <span id="err1"></span>
    </div>
    <div id="selectTrain" style="display: none">
        <div class="wizard-title">Select the train from the list </div>
        <div id="availableTrain"></div>
        <br>
        <div class="btn-container">
            <button id="goToSearch" class="e-btn">Back</button>
            <button id="bookTickets" class="e-btn">Continue</button>
        </div>
        <span id="err2"></span>
    </div>
    <div id="details" style="display: none">
        <div class="details-page wizard-title">Enter the passenger
details</div>
        <div id="PassengersList">
            <table id="passenger-table">
                <colgroup>
                    <col>
                    <col>
                    <col>
                    <col>
                    <col>
                    <col>
                </colgroup>
                <thead>
                    <tr>
                        <th class="name-header">Name</th>
                        <th class="age-header">Age</th>
                        <th class="gender-header">Gender</th>
                        <th class="type-header">Berth Preference</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>
                            <input id="pass_name1" class="e-input"
type="text" placeholder="Passenger Name">
                        </td>
                        <td>
                            <input id="pass_age1" class="e-input">
                        </td>
                        <td>
                            <input id="pass_gender1" type="text">
                        </td>
                        <td>
                            <input id="pass_berth1" type="text">
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <input id="pass_name2" class="e-input"
type="text" placeholder="Passenger Name">
                        </td>

```

```

        <td>
            <input id="pass_age2" class="e-input">
        </td>
        <td>
            <input id="pass_gender2" type="text">
        </td>
        <td>
            <input id="pass_berth2" type="text">
        </td>
    </tr>
    <tr>
        <td>
            <input id="pass_name3" class="e-input"
type="text" placeholder="Passenger Name">
        </td>
        <td>
            <input id="pass_age3" class="e-input">
        </td>
        <td>
            <input id="pass_gender3" type="text">
        </td>
        <td>
            <input id="pass_berth3" type="text">
        </td>
    </tr>
</tbody>
</table>
</div>
<br>
<div class="btn-container">
    <button id="goBackToBook" class="e-btn">Back</button>
    <button id="confirmTickets" class="e-btn">Continue</button>
</div>
<span id="err3"></span>
</div>
<div id="confirm" style="display: none">
    <div class="tab-title1 wizard-title">Confirm the details and
proceed</div>
    <div id="ticketDetailGrid"></div>
    <div id="amount"></div>
    <br>
    <div class="btn-container">
        <button id="goBackDetails" class="e-btn">Back</button>
        <button id="makePayment" class="e-btn">Pay</button>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Load tab with data source in EJ2 JavaScript Tab control

You can bind any data object to Tab items, by mapping it to a [header](#) and [content](#) property.

In the below demo, Data is fetched from an OData service using **DataManager**. The result data is formatted as a JSON object with **header** and **content** fields, which is set to items property of Tab.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
import { DataManager, Query, ODataV4Adaptor, ReturnOption } from
 '@syncfusion/ej2-data';
let itemsData: any = [];
let mapping = { header: 'FirstName', content: 'Notes' };
const SERVICE_URI: string =
 'https://services.odata.org/V4/Northwind/Northwind.svc/Employees';
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor,
crossDomain: true})
    .executeQuery(new Query().range(1, 4)).then((e: ReturnOption) => {
        let result: any = e.result;
        for(let i: number = 0; i < result.length; i++) {
            itemsData.push({ header: {text: result[i][mapping.header]}, content:
result[i][mapping.content] });
        }
        let tabObj: Tab = new Tab({
            items: itemsData
        });
        tabObj.appendTo('#element');
    });
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Tab">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
```

```

if(ele) {
    ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Add nested tabs in EJ2 JavaScript Tab control

Tab supports to render the nested level of Tabs by using `content` property. You can add the nested Tab element inside the parent Tab `content` property. To render the nested Tab, initialize the component using the id of Tab from a [selected](#) event handler.

INDEX.TS

```

import { Tab, SelectEventArgs } from '@syncfusion/ej2-navigations';
import { enableRipple, isNullOrUndefined as isNOU } from '@syncfusion/ej2-base';
import { usa_cities, france_cities, australia_cities } from './datasource.ts';
enableRipple(true);
let tabObj: Tab = new Tab({
    selected: handleSelectEvent,
    created: handleCreatedEvent,
    items: [
        {
            header: { 'text': 'USA' },
            content: '<div id="usa_tab"></div>'
        },
        {
            header: { 'text': 'France' },
            content: '<div id="france_tab"></div>'
        },
        {
            header: { 'text': 'Australia' },
            content: '<div id="australia_tab"></div>'
        }
    ]
});
tabObj.appendTo('#element');
function handleCreatedEvent(): void {
    if (isNOU(document.querySelector('#usa_tab.e-tab'))) {
        let usa_obj: Tab = new Tab({
            items: usa_cities
        });
        usa_obj.appendTo('#usa_tab');
    }
}
function handleSelectEvent(e: SelectEventArgs): void {
    if (e.selectedIndex === 0 && isNOU(document.querySelector('#usa_tab.e-tab'))) {
        let usa_obj: Tab = new Tab({
            items: usa_cities
        });
        usa_obj.appendTo('#usa_tab');
    } else if (e.selectedIndex === 1 && isNOU(document.querySelector('#france_tab.e-tab'))) {

```

```

let france_obj: Tab = new Tab({
  items: france_cities
});
france_obj.appendTo('#france_tab');
} else if (e.selectedIndex === 2 &&
isNOU(document.querySelector('#australia_tab.e-tab'))) {
  let australia_obj: Tab = new Tab({
    items: australia_cities
  });
  australia_obj.appendTo('#australia_tab');
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Tab">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Add font awesome in EJ2 JavaScript Tab control

We can customize the Tab component items by using font awesome icons and fonts.

- Need to add font awesome CDN reference link in your sample to use font awesome icons.


```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"/>
```

You can add the font awesome icons to the Tab component using '[iconCss](#)' property. The following sample illustrates how to use font awesome in tab component.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    items: [
        {
            header: { 'text': 'Twitter', 'iconCss': 'fa fa-twitter' },
            content: 'Twitter is an online social networking service that
enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and post
tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface, SMS
or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in July
2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets a
day in 2012. The service also handled 1.6 billion ' +
'search queries per day.'
        },
        {
            header: { 'text': 'Facebook', 'iconCss': 'fa fa-facebook' },
            content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The
founders had initially limited the website's ' +
'membership to Harvard students, but later expanded it to
colleges in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students at various
other universities and later to high-school students.'
        },
        {
            header: { 'text': 'WhatsApp', 'iconCss': 'fa fa-whatsapp' },
            content: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates ' +
'under a subscription business model. It uses the Internet to
send text messages, images, video, user location and ' +
'audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user ' +
'base of up to one billion, [10] making it the most globally
popular messaging application. WhatsApp Inc., based in ' +
'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.'
        }
    ]
});
```

```
tabObj.appendTo('#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Tab Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"> </div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Customization

Use the following CSS to customize the header item icon of Tab control.

```
`css

.e-tab .e-tab-header .e-toolbar-item .e-tab-icon {
color: #badfba !important;
}
`
```

Set state persistence of the tab component in EJ2 JavaScript Tab control

State persistence allows the Tab to retain the current modal value in the browser cookies for state maintenance. This action is handled through the [enablePersistence](#) property which is set to false by

default. When it is set to true, some of the Tab component model values will be retained even after refreshing the page.

The following sample demonstrates how to set state persistence of the Tab component.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    enablePersistence: true,
    items: [{
        header: { 'text': 'Twitter' },
        content: 'Twitter is an online social networking service that enables users to send and read short 140-character ' +
            'messages called "tweets". Registered users can read and post tweets, but those who are unregistered can only read ' +
            'them. Users access Twitter through the website interface, SMS or mobile device app Twitter Inc. is based in San ' +
            'Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, ' +
            'Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, ' +
            'with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion ' +
            'search queries per day.'
    },
    {
        header: { 'text': 'Facebook' },
        content: 'Facebook is an online social networking service headquartered in Menlo Park, California. Its website was ' +
            'launched on February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow students Eduardo ' +
            'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had initially limited the website's ' +
            'membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League, and Stanford ' +
            'University. It gradually added support for students at various other universities and later to high-school students.'
    },
    {
        header: { 'text': 'WhatsApp' },
        content: 'WhatsApp Messenger is a proprietary cross-platform instant messaging client for smartphones that operates ' +
            'under a subscription business model. It uses the Internet to send text messages, images, video, user location and ' +
            'audio media messages to other users using standard cellular mobile numbers. As of February 2016, WhatsApp had a user ' +
            'base of up to one billion, [10] making it the most globally popular messaging application. WhatsApp Inc., based in ' +
            'Mountain View, California, was acquired by Facebook Inc. on February 19, 2014, for approximately US$19.3 billion.'
    }
    ]
});
tabObj.appendTo('#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Tab">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Set custom animation in EJ2 JavaScript Tab control

Tab supports custom animations for both previous and next actions from the provided animation option of **Animation** library. The [animation](#) property also allows you to set [easing](#), [duration](#), and various other [effect](#).

Default animation is given as **SlideLeftIn** for [previous](#) tab animation and **SlideRightIn** for [next](#) tab animation. You can also disable the animation by setting the animation effect as **None**. Also, please use the following CSS to disable indicator animation for animation effect as **None**.

```
`css
```

```
.e-tab .e-tab-header:not(.e-vertical) .e-indicator, .e-tab .e-tab-header.e-vertical .e-indicator {
transition: none;
}
`
```

The sample demonstrates some types of animation that suits Tab. You can check all the animation effects [here](#).

INDEX.TS

```

import { Tab, selectEventArgs } from '@syncfusion/ej2-navigations';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
let tabObj: Tab = new Tab({
    items: [{
        header: { 'text': 'Twitter' },
        content: 'Twitter is an online social networking service that
enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and post
tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface, SMS
or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in July
2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets a
day in 2012. The service also handled 1.6 billion ' +
'search queries per day.'
    },
    {
        header: { 'text': 'Facebook' },
        content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The
founders had initially limited the website's ' +
'membership to Harvard students, but later expanded it to
colleges in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students at various
other universities and later to high-school students.'
    },
    {
        header: { 'text': 'WhatsApp' },
        content: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates ' +
'under a subscription business model. It uses the Internet to
send text messages, images, video, user location and ' +
'audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user ' +
'base of up to one billion, [10] making it the most globally
popular messaging application. WhatsApp Inc., based in ' +
'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.'
    }
    ]
});
tabObj.appendTo('#element');
let listObjPrevious: DropDownList = new DropDownList({
    index: 0,
    placeholder: 'Select a animate type',
    popupHeight: '150px',
    change: () => { valueChange(); }
});
listObjPrevious.appendTo('#previousAnimation');
let listObjNext: DropDownList = new DropDownList({
    index: 1,

```

```

        placeholder: 'Select a animate type',
        popupHeight: '150px',
        change: () => { valueChange1(); }
    });
    listObjNext.appendTo('#nextAnimation');
    function valueChange(): void {
        tabObj.animation.previous.effect = listObjPrevious.value;
    }
    function valueChange1(): void {
        tabObj.animation.next.effect = listObjNext.value;
    }

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 Tab">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="default" style="padding-bottom:75px;">
      <div class="row">
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
          <label> Previous Animation </label>
        </div>
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
          <select id="previousAnimation">
            <option value="SlideLeftIn">SlideLeftIn</option>
            <option value="SlideRightIn">SlideRightIn</option>
            <option value="FadeIn">FadeIn</option>
            <option value="FadeOut">FadeOut</option>
            <option value="FadeZoomIn">FadeZoomIn</option>
            <option value="FadeZoomOut">FadeZoomOut</option>
            <option value="ZoomIn">ZoomIn</option>
            <option value="ZoomOut">ZoomOut</option>
            <option value="None">None</option>
          </select>
        </div>
      </div>
      <div class="row">
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
          <label> Next Animation </label>
        </div>

```

```

        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
            <select id="nextAnimation">
                <option value="SlideLeftIn">SlideLeftIn</option>
                <option value="SlideRightIn">SlideRightIn</option>
                <option value="FadeIn">FadeIn</option>
                <option value="FadeOut">FadeOut</option>
                <option value="FadeZoomIn">FadeZoomIn</option>
                <option value="FadeZoomOut">FadeZoomOut</option>
                <option value="ZoomIn">ZoomIn</option>
                <option value="ZoomOut">ZoomOut</option>
                <option value="None">None</option>
            </select>
        </div>
    </div>
    <div id="element"></div>
    <br><br>
    <div id="result"></div>
</div><script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Load tab items dynamically in EJ2 JavaScript Tab control

Tabs can be added dynamically by passing array of items and index value to the [addTab](#) method.

```
`ts
```

```
// New tab title and content inputs are fetched and stored in local variable
```

```
let title: string = document.getElementById('tab-title').value;
```

```
let content: string = document.getElementById('tab-content').value;
```

```
// Required tab item object formed by using textbox inputs
```

```
let item: Object = { header: { text: title }, content: createElement('pre', { innerHTML:
content.replace(/\n/g, '<br>\n') }).outerHTML };
```

```
// Item object and the index argument passed into the addTab method to add a new tab
```

```
tabObj.addTab([item], index);
```

```
,
```

In the following demo, you can add the tab content by clicking the +. This + icon is added on the tab header using [iconCss](#) property. Enter the new Tab heading and content details in the available text boxes, click 'Add Tab' button to pass the details as an array and here last index is calculated to append the new tab at the end.

INDEX.TS

```
import { enableRipple, createElement } from '@syncfusion/ej2-base';
import { Tab, SelectEventArgs } from '@syncfusion/ej2-navigations';
```

```

enableRipple(true);
let totalItems: number = 0;
let tabObj: Tab = new Tab({
    selected: tabSelected,
    items: [
        {
            header: { 'text': 'Tab1' },
            content: '#tab1_content'
        },
        {
            header: { 'iconCss': 'e-add-icon' },
            content: '#form-container'
        }
    ]
});
tabObj.appendTo('#element');
let addBtn : HTMLElement[] = document.querySelectorAll(".e-ileft.e-icon");
addBtn[0].setAttribute("title", "Add Tab");
function tabSelected(args: SelectEventArgs): void {
    if (args.selectedIndex === document.querySelectorAll('#element .e-
toolbar-item').length - 1) {
        document.getElementById('tab-title').value = '';
        document.getElementById('tab-content').value = '';
    }
}
document.getElementById('btn-add').onclick = (e : Event) => {
    let title: string = document.getElementById('tab-title').value;
    let content: string = document.getElementById('tab-content').value;
    let item: Object = { header: { text: title }, content:
createElement('pre', { innerHTML: content.replace(/\n/g, '<br>\n')
}).outerHTML };
    totalItems = document.querySelectorAll('#element .e-toolbar-
item').length;
    tabObj.addTab([item], totalItems-1);
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Tab</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Tab">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

<div id="container">
  <div id="tab1_content" style="display: none">
    <ul>
      <li>Click on the "+" header to add dynamic tab items. </li>
      <li>It displays input elements to get the new tab
information. </li>
      <li>Add details and click the "Add Tab" button to open the
newly added tab.</li>
    </ul>
  </div>
  <div id="form-container" style="display: none">
    <div class="e-float-input">
      <input type="text" id="tab-title" required="">
      <span class="e-float-line"></span>
      <label class="e-float-text">Enter header title</label>
    </div>
    <br>
    <div class="e-float-input">
      <textarea rows="5" type="text" id="tab-content"
required=""></textarea>
      <span class="e-float-line"></span>
      <label class="e-float-text">Enter content</label>
    </div>
    <br>
    <div class="btn-section">
      <button id="btn-add" class="e-btn">Add Tab</button>
      <br>
      <br>
      <span class="info"> * Title is mandatory to add a new
Tab</span>
    </div>
  </div>
  <div id="element"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Create collapsible tabs in EJ2 JavaScript Tab control

You can achieve collapse and expand functionality in Tab by adding/removing a custom CSS class in the click event handler for each tab.

- Define a CSS class to set the style property display as none. Here 'collapse' class is added to the content element for hiding it using [created](#) event.
- Bind the [selected](#) event for Tab to collapse the initially selected Tab item and bind custom click handler for the Tab headers.
- In the event handler, add and remove 'collapse' class to hide and show the corresponding Tab content.

INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { Tab, SelectEventArgs } from '@syncfusion/ej2-navigations';
enableRipple(true);
let trgIndex: number;
let actLine: HTMLElement;
let tabObj: Tab = new Tab({
    created: tabCreated,
    selected: tabSelected,
    items: [
        {
            header: { 'text': 'Twitter' },
            content: 'Twitter is an online social networking service that
enables users to send and read short
+ '140-character messages called "tweets". Registered users can
read and post tweets, but those
+ 'who are unregistered can only read them. Users access Twitter
through the website interface, '
+ 'SMS or mobile device app Twitter Inc. is based in San
Francisco and has more than 25 offices '
+ 'around the world. Twitter was created in March 2006 by Jack
Dorsey, Evan Williams, Biz Stone, '
+ 'and Noah Glass and launched in July 2006. The service rapidly
gained worldwide popularity, '
+ 'with more than 100 million users posting 340 million tweets a
day in 2012. The service also '
+ 'handled 1.6 billion search queries per day.'
        },
        {
            header: { 'text': 'Facebook' },
            content: 'Facebook is an online social networking service
headquartered in Menlo Park, California.'
+ 'Its website was launched on February 4, 2004, by Mark
Zuckerberg with his Harvard College '
+ 'roommates and fellow students Eduardo Saverin, Andrew
McCollum, Dustin Moskovitz and Chris '
+ 'Hughes. The founders had initially limited the website\'\'s
membership to Harvard students, but '
+ 'later expanded it to colleges in the Boston area, the Ivy
League, and Stanford University. It '
+ 'gradually added support for students at various other
universities and later to high-school '
+ 'students.'
        },
        {
            header: { 'text': 'WhatsApp' },
            content: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for '
+ 'smartphones that operates under a subscription business
model. It uses the Internet to send '
+ 'text messages, images, video, user location and audio media
messages to other users using '
+ 'standard cellular mobile numbers. As of February 2016,
WhatsApp had a user base of up to one '
+ 'billion, [10] making it the most globally popular messaging
application. WhatsApp Inc., based in '

```

```

        + 'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for '
        + 'approximately US$19.3 billion.'
    }
    ]
});
//Render initialized Tab component
tabObj.appendTo('#element');
function tabCreated(): void {
    // After tab created first tab content and active line are hidden by
    // adding custom class to make it collapse state
    actLine = document.querySelector('.e-indicator');
    document.getElementById('e-content-
element_0').classList.add('collapse');
    actLine.classList.add('collapse');
}
function tabSelected(e: SelectEventArgs): void {
    // If next tab item selected custom class is removed from content and
    // active line element
    let cnttrgs: HTMLElement[] = document.querySelectorAll('#element.e-tab >
.e-content > .e-item');
    for (let i: number = 0; i < cnttrgs.length; i++) {
        cnttrgs[i].classList.remove('collapse');
    }
    if (actLine !== undefined) { actLine.classList.remove('collapse'); }
    this.trgIndex = e.selectedIndex;
    // Custom click event binding for each tab item to make collapse/expand
    e.selectedItem.onclick = (e : Event) => {
        updateCollapseClass(this.trgIndex);
    };
}
function updateCollapseClass(index: number): void {
    // Custom classes are added/removed from tab content and active line
    // element, when the same tab item again clicked
    let cntEle: HTMLElement = document.getElementById('e-content-' +
'element_' + index);
    if (cntEle.classList.contains('collapse')) {
        cntEle.classList.remove('collapse');
        actLine.classList.remove('collapse');
    } else {
        cntEle.classList.add('collapse');
        actLine.classList.add('collapse');
    }
}
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Tab</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 Tab">
    <meta name="author" content="Syncfusion">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
    <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="info">
            Collapsible Tabs
        </div>
        <span style="margin: 10px;">
            <i>The active tab can be toggled to expand and collapse its
content.</i>
        </span>
        <br><br>
        <div id="element" class="e-background"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize tab content height in EJ2 JavaScript Tab control

You can change the Tab content height by using the [heightAdjustMode](#) property. By default, the Tab content [heightAdjustMode](#) property is set to **Content** value.

- **None:** Each tab content height is set based on the Tab height. This value is used only the tab component having the [height](#) property.
- **Auto:** Each tab content height will take the maximum height of all other tabs content.
- **Content:** Each tab content height is set based on their own content.
- **Fill:** Each tab content height is set based on the full height of Tabs parent element.

INDEX.TS

```

import { Tab } from '@syncfusion/ej2-navigations';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-
dropdowns';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let tabObj: Tab = new Tab({
    height: "400px",
    items: [
        {
            header: { 'text': 'Twitter' },
            content: 'Twitter is an online social networking service
that enables users to send and read short 140-character ' +

```

```

        'messages called "tweets". Registered users can read and
        post tweets, but those who are unregistered can only read ' +
        'them. Users access Twitter through the website interface,
        SMS or mobile device app Twitter Inc. is based in San ' +
        'Francisco and has more than 25 offices around the world.
        Twitter was created in March 2006 by Jack Dorsey, ' +
        'Evan Williams, Biz Stone, and Noah Glass and launched in
        July 2006. The service rapidly gained worldwide popularity, ' +
        'with more than 100 million users posting 340 million tweets
        a day in 2012. The service also handled 1.6 billion ' +
        'search queries per day.'
    },
    {
        header: { 'text': 'Facebook' },
        content: 'Facebook is an online social networking service
        headquartered in Menlo Park, California. Its website was ' +
        'launched on February 4, 2004, by Mark Zuckerberg with his
        Harvard College roommates and fellow students Eduardo ' +
        'Saverin, Andrew McCollum, Dustin Moskovitz and Chris
        Hughes. The founders had initially limited the website's ' +
        'membership to Harvard students, but later expanded it to
        colleges in the Boston area, the Ivy League, and Stanford ' +
        'University. It gradually added support for students at
        various other universities and later to high-school students.'
    },
    {
        header: { 'text': 'WhatsApp' },
        content: 'WhatsApp Messenger is a proprietary cross-platform
        instant messaging client for smartphones that operates ' +
        'under a subscription business model. It uses the Internet
        to send text messages, images, video, user location and ' +
        'audio media messages to other users using standard cellular
        mobile numbers. As of February 2016, WhatsApp had a user ' +
        'base of up to one billion, [10] making it the most globally
        popular messaging application. WhatsApp Inc., based in ' +
        'Mountain View, California, was acquired by Facebook Inc. on
        February 19, 2014, for approximately US$19.3 billion.'
    }
]
});
tabObj.appendTo('#element');
let heightValues: DropDownList = new DropDownList({
    width: '90%',
    index: 1,
    change: () => { changeHeightValue(); }
});
heightValues.appendTo('#contentHeight');
function changeHeightValue(e: ChangeEventArgs): void {
    tabObj.heightAdjustMode = heightValues.value;
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>Essential JS 2 Tab</title>
<meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Toolbar Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div id="default">
            <div class="row">
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <label>Height Adjust Mode</label>
                </div>
                <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
                    <select id="contentHeight">
                        <option value="None">None</option>
                        <option value="Content">Content</option>
                        <option value="Fill">Fill</option>
                        <option value="Auto">Auto</option>
                    </select>
                </div>
            </div>
            <br>
            <div class="e-tab-height" style="height:300px">
                <div id="element"></div>
            </div>
            <div id="result"></div>
        </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Reorder active tab in EJ2 JavaScript Tab control

We can able to prevent the changing of the active tab item on resizing the browser when overflow mode is popup by using the [reorderActiveTab](#) property. By default, the active Tab should be reordered when we click the tab items from the popup. If we set false to [reorderActiveTab](#) property the active tab item from the popup will not be reordered and an active item is highlighted inside the popup. The following code example depicts to prevent the reorder active tab item inside the popup.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
  overflowMode: 'Popup',
  heightAdjustMode: 'Auto',
  reorderActiveTab: false,
  items: [
    {
      header: { 'text': 'India' },
      content: 'India officially the Republic of India, is a
country in South Asia. It is the seventh-largest country by area, the
second-most populous country with over 1.2 billion people, and the most
populous democracy in the world. Bounded by the Indian Ocean on the south,
the Arabian Sea on the south-west, and the Bay of Bengal on the south-east,
it shares land borders with Pakistan to the west;China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.'
    },
    {
      header: { 'text': 'Canada' },
      content: 'Canada is a North American country stretching from
the U.S. in the south to the Arctic Circle in the north. Major cities
include massive Toronto, west coast film centre Vancouver, French-speaking
Montréal and Québec City, and capital city Ottawa. Canada vast swaths of
wilderness include lake-filled Banff National Park in the Rocky Mountains.
It also home to Niagara Falls, a famous group of massive waterfalls.'
    },
    {
      header: { 'text': 'Australia' },
      content: 'Australia, officially the Commonwealth of
Australia, is a country comprising the mainland of the Australian continent,
the island of Tasmania and numerous smaller islands. It is the world sixth-
largest country by total area. Neighboring countries include Indonesia, East
Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and
New Caledonia to the north-east; and New Zealand to the south-east.
<br/><br/>India is a vast South Asian country with diverse terrain - from
Himalayan peaks to Indian Ocean coastline - and history reaching back 5
millennia. In the north, Mughal Empire and marks include Delhi\'s Red Fort
complex and massive Jama Masjid mosque, plus Agras iconic Taj Mahal
mausoleum. Pilgrims bathe in the Ganges in Varanasi, and Rishikesh is a yoga
centre and base for Himalayan trekking.'
    },
    {
      header: { 'text': 'USA' },
      content: 'The United States of America (USA or U.S.A.),
commonly called the United States (US or U.S.) and America, is a federal
republic consisting of fifty states and a federal district. The 48
contiguous states and the federal district of Washington, D.C. are in
central North America between Canada and Mexico. The state of Alaska is west
of Canada and east of Russia across the Bering Strait, and the state of
Hawaii is in the mid-North Pacific. The country also has five populated and
nine unpopulated territories in the Pacific and the Caribbean.'

```

```

        header: { 'text': 'London' },
        content: 'London, the capital of England and the United
Kingdom, is a 21st-century city with history stretching back to Roman times.
At its centre stand the imposing Houses of Parliament, the iconic 'Big Ben'
clock tower and Westminster Abbey, site of British monarch coronations.
Across the Thames River, the London Eye observation wheel provides panoramic
views of the South Bank cultural complex, and the entire city.'
    },
    {
        header: { 'text': 'Germany' },
        content: 'Germany is a Western European country with a
landscape of forests, rivers, mountain ranges and North Sea beaches. It has
over 2 millennia of history. Berlin, its capital, is home to art and
nightlife scenes, the Brandenburg Gate and many sites relating to WWII.
Munich is known for its Oktoberfest and beer halls, including the 16th-
century Hofbräuhaus. Frankfurt, with its skyscrapers, houses the European
Central Bank.'
    },
    {
        header: { 'text': 'France' },
        content: 'France, officially the French Republic is a
sovereign state comprising territory in western Europe and several overseas
regions and territories. The European part of France, called Metropolitan
France, extends from the Mediterranean Sea to the English Channel and the
North Sea, and from the Rhine to the Atlantic Ocean; France covers 640,679
square kilo metres and as of August 2015 has a population of 67 million,
counting all the overseas departments and territories.'
    },
    {
        header: { 'text': 'Sweden' },
        content: 'Sweden is a Scandinavian nation with thousands of
coastal islands and inland lakes, along with vast boreal forests and
glaciated mountains. Its principal cities, eastern capital Stockholm and
southwestern Gothenburg and Malmö, are all coastal. Stockholm is built on 14
islands. It has more than 50 bridges, as well as the medieval old town,
Gamla Stan, royal palaces and museums such as open-air Skansen.'
    },
    {
        header: { 'text': 'Africa' },
        content: 'Africa is the world second-largest and second-
most-populous continent. At about 30.3 million km2 including adjacent
islands, it covers 6% of Earth total surface area and 20.4% of its total
land area'
    },
    {
        header: { 'text': 'Japan' },
        content: 'Japan is an island nation in the Pacific Ocean
with dense cities, imperial palaces, mountainous national parks and
thousands of shrines and temples. Shinkansen bullet trains connect the main
islands of Kyushu, Honshu (home to Tokyo and Hiroshima's atomic-bomb
memorial) and Hokkaido (famous for skiing). Tokyo, the capital, is known for
skyscrapers, shopping and pop culture.'
    },
    {
        header: { 'text': 'Malaysia' },
        content: 'Malaysia is a Southeast Asian country occupying
parts of the Malay Peninsula and the island of Borneo. It known for its

```



```

beaches, rainforests and mix of Malay, Chinese, Indian and European cultural
influences. The capital, Kuala Lumpur, is home to colonial buildings, busy
shopping districts such as Bukit Bintang and skyscrapers such as the iconic,
451m-tall Petronas Twin Towers.'
    },
    {
        header: { 'text': 'Singapore' },
        content: 'Singapore, an island city-state off southern
Malaysia, is a global financial center with a tropical climate and
multicultural population. Its colonial core centers on the Padang, a cricket
field since the 1830s and now flanked by grand buildings such as City Hall,
with its 18 Corinthian columns. In Singapore circa-1820 Chinatown stands the
red-and-gold Buddha Tooth Relic Temple, said to house one of Buddha teeth.'
    }
]
});
tabObj.appendTo('#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-tab-height" style="height:300px">
      <div id="element"></div>
    </div>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Display tool tip on tab header in EJ2 JavaScript Tab control

You can display tooltip for the tab component header using [beforeRender](#) event of Essential JS 2 Tooltip component which can be viewed as hint texts on mouse hovers over tab.

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
import { Tooltip, TooltipEventArgs } from '@syncfusion/ej2-popups';
let tabObj: Tab = new Tab({
  heightAdjustMode: 'Auto',
  items: [
    {
      header: { 'text': 'Twitter' },
      content: 'Twitter is an online social networking service that
enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and post
tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface, SMS
or mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in July
2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets a
day in 2012. The service also handled 1.6 billion ' +
'search queries per day.'
    },
    {
      header: { 'text': 'Facebook' },
      content: 'Facebook is an online social networking service
headquartered in Menlo Park, California. Its website was ' +
'launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The
founders had initially limited the website\'s ' +
'membership to Harvard students, but later expanded it to
colleges in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students at various
other universities and later to high-school students.'
    },
    {
      header: { 'text': 'WhatsApp' },
      content: 'WhatsApp Messenger is a proprietary cross-platform
instant messaging client for smartphones that operates ' +
'under a subscription business model. It uses the Internet to
send text messages, images, video, user location and ' +
'audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user ' +
'base of up to one billion, [10] making it the most globally
popular messaging application. WhatsApp Inc., based in ' +
'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.'
    }
  ]
});
tabObj.appendTo('#element');
let tooltip: Tooltip = new Tooltip({
```

```

    target: '.e-toolbar-item',
    position: 'TopCenter',
    content: '',
    beforeRender: onBeforeRender
  });
  tooltip.appendTo('#Tooltip');
  function onBeforeRender(args: TooltipEventArgs): void {
    if (args.target) {
      this.content = `<div>` + args.target.innerText + `</div>`;
    }
  }
}

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Toolbar</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="Tooltip"><div id="element"></div></div>
    <br><br>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tab selection in EJ2 JavaScript Tab control

We can able to find the tab selection whether it is selected by user interaction or programmatically way in the [selecting](#) and [selected](#) event argument with the field of `isInteracted`. When the user changes the tab through click actions it will return `true` otherwise, it will return false. The following code example depicts to find the tab selecting the state in selecting and selected events.

INDEX.TS

```
import { Tab, SelectingEventArgs, SelectEventArgs } from '@syncfusion/ej2-
navigations';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
let tabObj: Tab = new Tab({
    heightAdjustMode: 'Auto',
    items: [
        {
            header: { 'text': 'India' },
            content: 'India officially the Republic of India, is a country
in South Asia. It is the seventh-largest country by area, the second-most
populous country with over 1.2 billion people, and the most populous
democracy in the world. Bounded by the Indian Ocean on the south, the
Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it
shares land borders with Pakistan to the west; China, Nepal, and Bhutan to
the north-east; and Burma and Bangladesh to the east. In the Indian Ocean,
India is in the vicinity of Sri Lanka and the Maldives; in addition, India
Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.'
        },
        {
            header: { 'text': 'Canada' },
            content: 'Canada is a North American country stretching from the
U.S. in the south to the Arctic Circle in the north. Major cities include
massive Toronto, west coast film centre Vancouver, French-speaking Montréal
and Québec City, and capital city Ottawa. Canada vast swaths of wilderness
include lake-filled Banff National Park in the Rocky Mountains. It also home
to Niagara Falls, a famous group of massive waterfalls.'
        },
        {
            header: { 'text': 'Australia' },
            content: 'Australia, officially the Commonwealth of Australia,
is a country comprising the mainland of the Australian continent, the island
of Tasmania and numerous smaller islands. It is the world sixth-largest
country by total area. Neighboring countries include Indonesia, East Timor
and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New
Caledonia to the north-east; and New Zealand to the south-east.
<br/><br/>India is a vast South Asian country with diverse terrain - from
Himalayan peaks to Indian Ocean coastline - and history reaching back 5
millennia. In the north, Mughal Empire landmarks include Delhi's Red Fort
complex and massive Jama Masjid mosque, plus Agra's iconic Taj Mahal
mausoleum. Pilgrims bathe in the Ganges in Varanasi, and Rishikesh is a yoga
centre and base for Himalayan trekking.'
        },
        {
            header: { 'text': 'USA' },
            content: 'The United States of America (USA or U.S.A.), commonly
called the United States (US or U.S.) and America, is a federal republic
consisting of fifty states and a federal district. The 48 contiguous states
and the federal district of Washington, D.C. are in central North America
between Canada and Mexico. The state of Alaska is west of Canada and east of
Russia across the Bering Strait, and the state of Hawaii is in the mid-North
Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.'

```

```

        header: { 'text': 'London' },
        content: 'London, the capital of England and the United Kingdom,
is a 21st-century city with history stretching back to Roman times. At its
centre stand the imposing Houses of Parliament, the iconic 'Big Ben' clock
tower and Westminster Abbey, site of British monarch coronations. Across the
Thames River, the London Eye observation wheel provides panoramic views of
the South Bank cultural complex, and the entire city.'
    },
    {
        header: { 'text': 'Germany' },
        content: 'Germany is a Western European country with a landscape
of forests, rivers, mountain ranges and North Sea beaches. It has over 2
millennia of history. Berlin, its capital, is home to art and nightlife
scenes, the Brandenburg Gate and many sites relating to WWII. Munich is
known for its Oktoberfest and beer halls, including the 16th-century
Hofbräuhaus. Frankfurt, with its skyscrapers, houses the European Central
Bank.'
    },
    {
        header: { 'text': 'France' },
        content: 'France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.'
    }
],
selecting: (args: SelectingEventArgs) => {
    getInteractionDetail(args.isInteracted);
},
selected: (args: SelectEventArgs) => {
    getInteractionDetail(args.isInteracted);
}
});
tabObj.appendTo('#element');
/**
 * @param htmlContent
 */
function getInteractionDetail(interact: boolean): void {
    let eventlog = interact ? 'Tab Item selected by user interaction'
    : 'Tab Item selected by programmatically';
    document.getElementById('EventLog').innerHTML =
document.getElementById('EventLog').innerHTML + '<b>' + eventlog +
'</b></span>';
}
var dropdownData = [
    { text: 'India', value: 0 },
    { text: 'Canada', value: 1 },
    { text: 'Australia', value: 2 },
    { text: 'USA', value: 3 },
    { text: 'London', value: 4 },
    { text: 'Germany', value: 5 },
    { text: 'France', value: 6 },
];
let selctItemObj: DropDownList = new DropDownList({

```

```

    index: 0,
    dataSource: dropdownData,
    placeholder: 'Select Tab Item using dropdown',
    floatLabelType: "Always",
    popupHeight: '150px',
    change: (args) => {
        tabObj.select(args.itemData.value);
    },
  });
  selctItemObj.appendTo('#selectTab');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-tab-height" style="height:300px">
      <div class="EventLog" id="EventLog" style="word-break:
normal;padding: 5px;"></div>
      <div id="selectTab"></div>
      <div id="element"></div>
    </div>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

Tab key navigation in EJ2 JavaScript Tab control

The [tabIndex](#) property of a Tab item is used to enable tab key navigation for that particular item. When a positive value is assigned to the [tabIndex](#) property, it allows the user to switch focus to the next or

previous tab item using the Tab or Shift+Tab keys. By default, the user can only switch between tab items using the arrow keys.

If the [tabIndex](#) value is set to 0 for all tab items, the tab will switch based on the order of the elements on the page. This means that if the tab items are listed in a specific order on the page, the user will be able to navigate through them using the Tab key in that same order.

To use the [tabIndex](#) property, you can assign a positive value to the property of each tab item that you want to enable tab key navigation. For example:

INDEX.TS

```
import { Tab } from '@syncfusion/ej2-navigations';
let tabObj: Tab = new Tab({
    heightAdjustMode: 'Auto',
    overflowMode: 'Scrollable',
    items: [
        {
            header: { 'text': 'HTML' }, tabIndex : 0,
            content: 'HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web ' +
'pages. Along with CSS, and JavaScript, HTML is a cornerstone
technology, used by most websites to create visually ' +
'engaging web pages, user interfaces for web applications, and
user interfaces for many mobile applications.[1] Web ' +
'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
'website semantically along with cues for presentation, making
it a markup language, rather than a programming language.'
        },
        {
            header: { 'text': 'C Sharp(C#)' }, tabIndex : 0,
            content: 'C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its development ' +
'team is led by Anders Hejlsberg. The most recent version is C#
5.0, which was released on August 15, 2012.'
        },
        {
            header: { 'text': 'Java' }, tabIndex : 0,
            content: 'Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle ' +
'Corporation, that provides a system for developing application
software and deploying it in a cross-platform computing ' +
'environment. Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to ' +
'enterprise servers and supercomputers. While less common, Java
applets run in secure, sandboxed environments to ' +
'provide many features of native applications and can be
embedded in HTML pages.'
        },
        {
            header: { 'text': 'VB.Net' }, tabIndex : 0,
            content: 'The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also ' +
'includes a command-line VB.NET compiler. The most recent
version is VB 2012, which was released on August 15, 2012.'
        },
    ]
});
```

```

        {
            header: { 'text': 'Xamarin' }, tabIndex : 0,
            content: 'Xamarin is a San Francisco, California based software
company created in May 2011[3] by the engineers that ' +
                'created Mono,[4] Mono for Android and MonoTouch that are cross-
platform implementations of the Common Language ' +
                'Infrastructure (CLI) and Common Language Specifications (often
called Microsoft .NET). With a C#-shared codebase, ' +
                'developers can use Xamarin tools to write native Android, iOS,
and Windows apps with native user interfaces and share ' +
                'code across multiple platforms.[5] Xamarin has over 1 million
developers in more than 120 countries around the World ' +
                'as of May 2015.'
        },
        {
            header: { 'text': 'ASP.NET' }, tabIndex : 0,
            content: 'ASP.NET is an open-source server-side web application
framework designed for web development to produce ' +
                'dynamic web pages. It was developed by Microsoft to allow
programmers to build dynamic web sites, web applications ' +
                'and web services. It was first released in January 2002 with
version 1.0 of the .NET Framework, and is the successor ' +
                'to Microsoft\\\'s Active Server Pages (ASP) technology. ASP.NET
is built on the Common Language Runtime (CLR), allowing ' +
                'programmers to write ASP.NET code using any supported .NET
language. The ASP.NET SOAP extension framework allows ' +
                'ASP.NET components to process SOAP messages.'
        },
        {
            header: { 'text': 'ASP.NET MVC' }, tabIndex : 0,
            content: 'The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the ' +
                'model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is ' +
                'proprietary. In the later versions of ASP.NET, ASP.NET MVC,
ASP.NET Web API, and ASP.NET Web Pages (a platform using ' +
                'only Razor pages) will merge into a unified MVC 6.The project
is called ASP.NET vNext.'
        },
        {
            header: { 'text': 'JavaScript' }, tabIndex : 0,
            content: 'JavaScript (JS) is an interpreted computer programming
language. It was originally implemented as part of ' +
                'web browsers so that client-side scripts could interact with
the user, control the browser, communicate ' +
                'asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in ' +
                'both game development and the creation of desktop
applications.'
        }
    ]
});
tabObj.appendTo('#element');

```

INDEX.HTML


```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Tab</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript Toolbar Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div class="e-tab-height" style="height:300px">
      <div class="EventLog" id="EventLog" style="word-break:
normal;padding: 5px;"></div>
      <div id="selectTab"></div>
      <div id="element"></div>
    </div>
    <div id="result"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

With this code, the user will be able to switch between the tab items using the Tab and Shift+Tab keys, in the order specified by the [tabIndex](#) values.

It's important to note that the [tabIndex](#) property only affects the ability to navigate between tab items using the Tab key. The user will still be able to use the arrow keys to switch between tab items, regardless of the value of the [tabIndex](#) property.

TextBox

Groups in EJ2 JavaScript Textbox control

The following section explains you the steps required to create TextBox with **icon** and **floating label**.

TextBox:

- Create a parent div element with the class **e-input-group**

- Place input element with the class `e-input` inside the parent div element.

```
<div class="e-input-group">
<input class="e-input" name='input' type="text" placeholder="Enter Date"/>
</div>
```

Floating label:

- Add the `e-float-input` class to the parent div element.
- Remove the `e-input` class and add `required` attribute to the input element.
- Place the span element with class `e-float-line` after the input element.
- Place the label element with class `e-float-text` after the above created span element. When you focus or filled with value in the TextBox, the label floats above the TextBox.

Creating the Floating label TextBox, you have to set the `required` attribute to the Input element to achieve the floating label functionality which is used for validating the value existence in TextBox. If you want to render the Floating label TextBox without

`required` attribute then refer to the [Floating label without required attribute](#) section.

```
<div class="e-float-input e-input-group">
<input type="text" required/>
<span class="e-float-line"></span>
<label class="e-float-text">Enter Name </label>
</div>
```

And refer to the following sections to add the icons to the TextBox.

With icon and floating label

Create an icon element as a span with the class `e-input-group-icon`, and the user can place the icon in either side of TextBox by adding the created icon element before/after the input.

For the floating label enabled TextBox add the icon element as first or last element inside the TextBox wrapper, and based on the element position it will act as prefix or suffix icon.

INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
    inputElement[i].addEventListener("focus", function () {
```

```

        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.add('e-input-focus');
        } else {
            this.parentNode.classList.add('e-input-focus');
        }
    });
    inputElement[i].addEventListener("blur", function () {
        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.remove('e-input-focus');
        } else {
            this.parentNode.classList.remove('e-input-focus');
        }
    });
}
// Add 'e-input-btn-ripple' class to the icon element for achieve ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (let i = 0; i < inputIcon.length; i++) {
    inputIcon[i].addEventListener('mousedown', function () {
        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        let element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Essential JS 2 TextBox</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 TextBox Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <script src="systemjs.config.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id='loader'>Loading....</div>
    <div id='container'>

```

```

        <div class='wrap'>
            <div id="input-container">
                <h4> TextBox with icons </h4>
                <div class="e-input-group">
                    <input class="e-input" type="text" placeholder="Enter Date"/>
                    <span class="e-input-group-icon e-input-popup-date"></span>
                </div>
                <div class="e-input-group e-float-icon-left">
                    <span class="e-input-group-icon e-input-date"></span>
                    <div class="e-input-in-wrap">
                        <input class="e-input" type="text" placeholder="Enter Date"/>
                    </div>
                </div>
                <div class="e-input-group e-float-icon-left">
                    <span class="e-input-group-icon e-input-date"></span>
                    <div class="e-input-in-wrap">
                        <input class="e-input" type="text" placeholder="Enter Date"/>
                        <span class="e-input-group-icon e-input-down"></span>
                    </div>
                </div>
                <h4> Floating label with icons </h4>
                <div class="e-float-input e-input-group">
                    <input required type="text" />
                    <span class="e-float-line"></span>
                    <label class="e-float-text"> Enter Date </label>
                    <span class="e-input-group-icon e-input-popup-date"></span>
                </div>
                <div class="e-float-input e-input-group e-float-icon-left">
                    <span class="e-input-group-icon e-input-date"></span>
                    <div class="e-input-in-wrap">
                        <input required type="text" />
                        <span class="e-float-line"></span>
                        <label class="e-float-text"> Enter Date </label>
                    </div>
                </div>
                <div class="e-float-input e-input-group e-float-icon-left">
                    <span class="e-input-group-icon e-input-date"></span>
                    <div class="e-input-in-wrap">
                        <input required type="text" />
                        <span class="e-float-line"></span>
                        <label class="e-float-text"> Enter Date </label>
                        <span class="e-input-group-icon e-input-down"></span>
                    </div>
                </div>
            </div>
            </div>
            </div>
            </div>
            <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
            </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
}
.e-input-group-icon:before {
    font-family: e-icons;
}
.e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow: adjoining-classes */
    font-size: 16px;
}
.e-input-group.e-small .e-input-group-icon.e-input-popup-date { /* csslint allow: adjoining-classes */
    font-size: 14px;
}
.e-input-group-icon.e-input-popup-date:before { /* csslint allow: adjoining-classes */
    content: "□";
}
.e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-classes */
    content: '\e85e';
}
.e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-classes */
    content: "□";
}
.e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-classes */
    content: '\e7ba';
}
.e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-classes */
    content: '\e814';
}
.e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-classes */
    content: "□";
}
```

```
.e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-classes */
  content: '\e904';
}
.e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-classes */
  content: '\e913';
}
.e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-classes */
  content: '\e837';
}
.e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-classes */
  content: '\e806';
}
#input-container .e-input-group { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
```

To place the icon on left side of the TextBox, create a div element with the class `e-input-in-wrap` as wrapper to the input element and place the floating line, floating text, and right side icon element within it. Add the `e-float-icon-left` class to the TextBox container element.

With clear button and floating label

The clear button is added to the input for clearing the value given in the TextBox. It is shown only when the input field has a value, otherwise not shown.

You can add the clear button to the TextBox by enabling the [showClearButton](#) API in textbox

INDEX.TS

```
import { TextBox } from '@syncfusion/ej2-inputs'
let inputobj: TextBox = new TextBox({
  placeholder: 'First Name',
  floatLabelType: 'Never'
});
inputobj.appendTo('#firstName');
let inputobj1: TextBox = new TextBox({
  placeholder: 'Last Name',
  floatLabelType: 'Auto'
});
inputobj1.appendTo('#lastName');
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Essential JS 2 TextBox</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```

<meta name="description" content="Essential JS 2 TextBox Components" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
<script src="systemjs.config.js"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id='loader'>Loading....</div>
<div id='container'>
<div class='wrap'>
<div id="input-container" class="textboxes">
<h4> Textbox with clear icon</h4>
<input id="firstName"/>
</div>
<div id="input-container" class="textboxes">
<h4>Floating Textbox with clear icon</h4>
<input id="lastName"/>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
visibility: hidden;
}
#loader {
color: #008cff;
font-family: 'Helvetica Neue', 'calibiri';
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
box-sizing: border-box;
margin: 0 auto;
padding: 30px 10px;
}

```

```
width: 260px;
}
.textboxes {
  margin-top: 15px;
}
```

Floating label without required attribute

You can render the **Floating label TextBox** without **required** attribute by manually float the label above of the TextBox using input events.

You can manually float the label above of the TextBox by adding the below list of classes to the floating label element. The classes are:

Class Name | Description

e-label-top | Floats the label above of the TextBox.

e-label-bottom | Label to be placed as placeholder for the TextBox.

INDEX.TS

```
/* To get the Input element */
let inputElement = document.getElementById('inpt1');
/* Update the label position based on Input value */
updateLabelState(inputElement.value,
inputElement.parentElement.querySelector('.e-float-text'));
inputElement.addEventListener("focus", function () {
  let label = this.parentElement.querySelector('.e-float-text');
  label.classList.add('e-label-bottom');
  label.classList.remove('e-label-top');
});
inputElement.addEventListener("blur", function () {
  updateLabelState(this.value, this.parentElement.querySelector('.e-float-
text'));
});
inputElement.addEventListener("input", function () {
  updateLabelState(this.value, this.parentElement.querySelector('.e-float-
text'));
});
/* Update the label position based on Input value */
/* e-label-top - Floats the label above of the TextBox */
/* e-label-bottom - Label to be placed as placeholder for the TextBox */
function updateLabelState(value, label) {
  if (value) {
    label.classList.add('e-label-top');
    label.classList.remove('e-label-bottom');
  } else {
    label.classList.add('e-label-bottom');
    label.classList.remove('e-label-top');
  }
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 TextBox</title>
```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 TextBox Components">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="input-container">
                <h4> Floating label without required attribute </h4>
                <div class="e-float-input">
                    <input type="text" id="inpt1">
                    <span class="e-float-line"></span>
                    <label class="e-float-text"> Enter Value </label>
                </div>
            </div>
        </div>
    </div>

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;

```

```
margin: 0 auto;
padding: 20px 10px;
width: 260px;
}
```

Multi-line input with floating label

Add the HTML textarea element with the **e-input** class to create default multi-line input.

Add the floating label support to the **multi-line input** by creating the floating label structure as defined in the initial section.

INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
    inputElement[i].addEventListener('focus', () => {
        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.add('e-input-focus');
        } else {
            this.parentNode.classList.add('e-input-focus');
        }
    });
    inputElement[i].addEventListener('blur', () => {
        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.remove('e-input-focus');
        } else {
            this.parentNode.classList.remove('e-input-focus');
        }
    });
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Essential JS 2 TextBox</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 TextBox Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
```

```

<script src="systemjs.config.js"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id='loader'>Loading....</div>
  <div id='container'>
    <div class='wrap'>
      <div id="input-container">
        <textarea class="e-input"> Address </textarea>
        <div class="e-float-input">
          <textarea required></textarea>
          <span class="e-float-line"></span>
          <label class="e-float-text"> Address </label>
        </div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  margin: 0 auto;
  padding: 20px 10px;
  width: 260px;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}

```

See Also

- [How to add floating label to TextBox programmatically](#)

- [Change the floating label color of the TextBox](#)
- [Change the color of the TextBox based on its value](#)

Sizing in EJ2 JavaScript Textbox control

You can render the TextBox in two different sizes.

Property | Description

Normal | By default, the TextBox is rendered with normal size.

Small | You need to add `e-small` class to the input element, or else add to the input container.

INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
    inputElement[i].addEventListener("focus", function () {
        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.add('e-input-focus');
        } else {
            this.parentNode.classList.add('e-input-focus');
        }
    });
    inputElement[i].addEventListener("blur", function () {
        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.remove('e-input-focus');
        } else {
            this.parentNode.classList.remove('e-input-focus');
        }
    });
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (let i = 0; i < inputIcon.length; i++) {
    inputIcon[i].addEventListener('mousedown', function () {
        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        let element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
```

```

<head>
  <title>Essential JS 2 TextBox</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 TextBox Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
  <script src="systemjs.config.js"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id='loader'>Loading....</div>
  <div id='container'>
    <div class='wrap'>
      <div id="input-container">
        <h4> Normal Size </h4>
        <div class="e-input-group">
          <input class="e-input" type="text" placeholder="Enter Name" />
        </div>
        <div class="e-float-input">
          <input type='text' required />
          <span class="e-float-line"></span>
          <label class="e-float-text">Enter Name</label>
        </div>
        <div class="e-input-group">
          <input class="e-input" type="text" placeholder="Enter Date" />
          <span class="e-input-group-icon e-input-popup-date"></span>
        </div>
        <h4> Small Size </h4>
        <div class="e-input-group e-small">
          <input class="e-input" type="text" placeholder="Enter Name" />
        </div>
        <div class="e-float-input e-small">
          <input type='text' required />
          <span class="e-float-line"></span>
          <label class="e-float-text">Enter Name</label>
        </div>
        <div class="e-input-group e-small">
          <input class="e-input" type="text" placeholder="Enter Date" />
          <span class="e-input-group-icon e-input-popup-date"></span>
        </div>
      </div>
    </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}

```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
}
.e-input-group-icon:before {
    font-family: e-icons;
}
.e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow: adjoining-classes */
    font-size:16px;
}
.e-input-group.e-small .e-input-group-icon.e-input-popup-date { /* csslint allow: adjoining-classes */
    font-size:14px;
}
.e-input-group-icon.e-input-popup-date:before { /* csslint allow: adjoining-classes */
    content: "□";
}
.e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-classes */
    content: '\e85e';
}
.e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-classes */
    content: "□";
}
.e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-classes */
    content: '\e7ba';
}
.e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-classes */
    content: '\e814';
}
```

```
.e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-classes */
  content: "□";
}
.e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-classes */
  content: '\e904';
}
.e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-classes */
  content: '\e913';
}
.e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-classes */
  content: '\e837';
}
.e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-classes */
  content: '\e806';
}
#input-container .e-input-group { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
```

Validation in EJ2 JavaScript Textbox control

The TextBox supports three types of validation styles namely **error**, **warning**, and **success**. These states are enabled by adding corresponding classes **.e-error**, **.e-warning**, or **.e-success** to the input parent element.

INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
  inputElement[i].addEventListener("focus", function () {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.add('e-input-focus');
    } else {
      this.parentNode.classList.add('e-input-focus');
    }
  });
  inputElement[i].addEventListener("blur", function () {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.remove('e-input-focus');
    } else {
      this.parentNode.classList.remove('e-input-focus');
    }
  });
}
```

```

    });
}
// Add 'e-input-btn-ripple' class to the icon element for achieve ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (let i = 0; i < inputIcon.length; i++) {
    inputIcon[i].addEventListener('mousedown', function () {
        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        let element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Essential JS 2 TextBox</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 TextBox Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <script src="systemjs.config.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <div class='wrap'>
            <div id="input-container">
                <div class="e-input-group e-warning">
                    <input class="e-input" type="text" placeholder="Input with warning" />
                </div>
                <div class="e-input-group e-error">
                    <input class="e-input" type="text" placeholder="Input with error" />
                </div>
                <div class="e-input-group e-success">
                    <input class="e-input" type="text" placeholder="Input with success" />
                </div>
            </div>
        </div>
    </div>

```



```
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
}
.e-input-group-icon:before {
    font-family: e-icons;
}
.e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow: adjoining-classes */
    font-size: 16px;
}
.e-input-group.e-small .e-input-group-icon.e-input-popup-date { /* csslint allow: adjoining-classes */
    font-size: 14px;
}
.e-input-group-icon.e-input-popup-date:before { /* csslint allow: adjoining-classes */
    content: "□";
}
.e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-classes */
    content: '\e85e';
}
.e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-classes */
    content: "□";
}
.e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-classes */
    content: '\e7ba';
}
```

```

}
.e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-
classes */
  content: '\e814';
}
.e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-
classes */
  content: "□";
}
.e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-
classes */
  content: '\e904';
}
.e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-
classes */
  content: '\e913';
}
.e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-
classes */
  content: '\e837';
}
.e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-
classes */
  content: '\e806';
}
#input-container .e-input-group { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}

```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using **.e-float-input.e-control-wrapper .e-float-text::after** class.

INDEX.TS

```

// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input,.e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
  inputElement[i].addEventListener("focus", function () {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.add('e-input-focus');
    } else {
      this.parentNode.classList.add('e-input-focus');
    }
  });
  inputElement[i].addEventListener("blur", function () {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.remove('e-input-focus');
    }
  });
}

```

```

    } else {
        this.parentNode.classList.remove('e-input-focus');
    }
});
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (let i = 0; i < inputIcon.length; i++) {
    inputIcon[i].addEventListener('mousedown', function () {
        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        let element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Essential JS 2 TextBox</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 TextBox Components" />
    <meta name="author" content="Syncfusion" />
    <link href="asterisk.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <script src="systemjs.config.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <div class='wrap'>
            <div id="input-container">
                <input class="e-input" type="text" placeholder="Enter Date"
floatLabelType="Auto"/>
                <div class="e-input-group">
                    <input class="e-input" name='input' type="text"
placeholder="Enter Date" floatLabelType="Auto"/>
                    <span class="e-input-group-icon e-input-popup-date"></span>
                </div>
            </div>
        </div>
    </div>

```

```

    </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
}
.e-float-input.e-control-wrapper .e-float-text::after {
    content: '*';
    color: red;
}

```

Multiline in EJ2 JavaScript Textbox control

This feature allows the textbox to accept one or more lines of text like address, description, comments, and more.

Create multiline textbox

You can convert the default textbox into the multiline textbox by setting the [multiline](#) API value as true or pass HTML5 textarea as element to the textbox.

The multiline textbox allows you to resize it in vertical direction alone.

INDEX.TS

```

import { TextBox } from '@syncfusion/ej2-inputs';
// Initialize TextBox component
let textareaObject: TextBox = new TextBox({
    placeholder: 'Enter your address',
    value: 'Mr. Dodsworth Dodsworth, System Analyst, Studio 103, The
    Business Center'
});

```

```
// Render initialized TextBox
textareaObject.appendTo('#default');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Multiline</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 TextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="multiline">
    <textarea id="default" name="text"></textarea>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
.multiline{
  margin: 30px auto;
  width: 30%;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
```

```
}
```

Implementing floating label

You can achieve the floating label behavior in the multiline textbox by setting [floatLabelType](#) as 'Auto'. The placeholder text act as floating label to the multiline textbox. You can provide the placeholder text to the multiline textbox either by using the [placeholder](#) property or placeholder attribute.

INDEX.TS

```
import { TextBox } from '@syncfusion/ej2-inputs';
// Initialize TextBox component with floating type as auto
let floatTypeAuto: TextBox = new TextBox({
    placeholder: 'Enter your address',
    floatLabelType: 'Auto'
});
// Render initialized floating type as auto TextBox
floatTypeAuto.appendTo('#multiline-auto');
// Initialize TextBox component with floating type as always
let floatTypeAlways: TextBox = new TextBox({
    placeholder: 'Enter your address',
    floatLabelType: 'Always'
});
// Render initialized floating type as always TextBox
floatTypeAlways.appendTo('#multiline-always');
// Initialize TextBox component with floating type as never
let floatTypeNever: TextBox = new TextBox({
    placeholder: 'Enter your address',
    floatLabelType: 'Never'
});
// Render initialized floating type as never TextBox
floatTypeNever.appendTo('#multiline-never');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Multiline</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 TextBox Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head>
<body>

    <label class="custom-input-label"> Float label type auto </label>
    <div id="container" class="multiline">
        <textarea id="multiline-auto" name="typeAuto"></textarea>
    </div>
    <label class="custom-input-label"> Float label type always </label>
    <div id="container1" class="multiline">
        <textarea id="multiline-always" name="typeAlways"></textarea>
    </div>
    <label class="custom-input-label"> Float label type never </label>
    <div id="container2" class="multiline">
        <textarea id="multiline-never" name="typeNever"></textarea>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.multiline{
    margin: 10px auto;
    width: 30%;
}
.custom-input-label {
    font-size: 14px;
    font-weight: bold;
    margin: 35%;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Auto resizing

By default, you can manually resize the multiline textbox. But you can also create an **auto resizing** multiline textbox with both the initial and dynamic value change. It can be done by calculating the height of the textarea in the created event for initial value update and in the input event for dynamic value update of the auto resize multiline textbox, as explained in the following code sample.

INDEX.TS

```

import { TextBox } from '@syncfusion/ej2-inputs';

```

```
// Initialize TextBox component
let autoResize: TextBox = new TextBox({
  placeholder: 'Enter your address',
  floatLabelType: 'Auto',
  value: "Mr. Dodsworth Dodsworth, System Analyst, Studio 103, The
Business Center",
  created: (args: any)=> {
    autoResize.addAttributes({rows: "1"});
    autoResize.element.style.height = "auto";
    autoResize.element.style.height = (autoResize.element.scrollHeight-
7) + "px";
  },
  input: (args: any)=> {
    args.event.currentTarget.style.height = "auto";
    args.event.currentTarget.style.height =
(args.event.currentTarget.scrollHeight)+"px";
  }
});
// Render initialized TextBox
autoResize.appendTo('#default');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Multiline</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 TextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="multiline">
    <textarea id="default" name="text"></textarea>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
.multiline{
    margin: 30px auto;
    width: 30%;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Disable resizing

By default, the multiline textbox is rendered with resizable. You can disable the resize of the multiline textbox by applying the following CSS styles.

```
`css
textarea.e-input,
.e-float-input textarea,
.e-float-input.e-control-wrapper textarea,
.e-input-group textarea,
.e-input-group.e-control-wrapper textarea {
    resize: none;
}
```

INDEX.TS

```
import { TextBox } from '@syncfusion/ej2-inputs';
// Initialize TextBox component
let disableResize: TextBox = new TextBox({
    placeholder: 'Enter your address',
    floatLabelType: 'Auto'
});
// Render initialized TextBox
disableResize.appendTo('#default');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Multiline</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 TextBox Component">
<meta name="author" content="Syncfusion">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container" class="multiline">
        <textarea id="default" name="text"></textarea>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.multiline{
    margin: 30px auto;
    width: 30%;
}
textarea.e-input,
.e-float-input textarea,
.e-float-input.e-control-wrapper textarea,
.e-input-group textarea,
.e-input-group.e-control-wrapper textarea {
    resize: none;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

```
}
```

Limit the text length

By default, the text length of the multiline textbox is unlimited. You can limit the text length by setting the `maxLength` attribute using the [addAttributes](#) method.

INDEX.TS

```
import { TextBox } from '@syncfusion/ej2-inputs';
import { Button } from '@syncfusion/ej2-buttons';
// Initialize TextBox component
let maxLength: TextBox = new TextBox({
    placeholder: 'Enter your address',
    floatLabelType: 'Auto'
});
// Render initialized TextBox
maxLength.appendTo('#default');
// Initialize TextBox component
let addAttr: TextBox = new TextBox({
    placeholder: 'Enter your address',
    floatLabelType: 'Auto'
});
// Render initialized TextBox
addAttr.appendTo('#attr');
// Initialize Button component
let button: Button = new Button({
});
// Render initialized Button
button.appendTo('#button');
document.getElementById('button').onclick = (): void => {
    addAttr.addAttributes({maxlength: 15});
}
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 Multiline</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 TextBox Component">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

    <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <label class="custom-input-label"> Add max length attribute through
inline </label>
    <div id="container" class="multiline">
        <textarea id="default" maxlength="15" name="text"></textarea>
    </div>
    <label class="custom-input-label"> Add maxlength attribute through
addAttributes method</label>
    <div id="sample" class="multiline">
        <textarea id="attr" name="length"></textarea>
    </div>
    <button id="button">Add max length</button>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.multiline{
    margin-top: 10px;
    margin-left: 170px;
    width: 30%;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
button{
    margin-left: 170px;
}
.custom-input-label {
    font-size: 14px;
    font-weight: bold;
    margin: 170px;
}

```

Count characters

You can show the number of characters entered inside the textarea by calculating the character count in the input event of multiline textbox. The character count is updated while entering or deleting any

character inside the textarea. The character count shows how many characters can be entered or left to be entered.

INDEX.TS

```
import { TextBox } from '@syncfusion/ej2-inputs';
// Initialize TextBox component
let countChar: TextBox = new TextBox({
  placeholder: 'Enter your address',
  floatLabelType: 'Auto',
  input: (args: any)=> {
    let word, addresscountRem, addressCount;
    word = args.value;
    addressCount = word.length;
    addresscountRem = document.getElementById('numbercount');
    addresscountRem.textContent = addressCount+"/25";
  }
});
// Render initialized TextBox
countChar.appendTo('#default');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Multiline</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 TextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container" class="multiline">
    <textarea id="default" maxlength="25" name="text"></textarea>
    <div id="numbercount"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.multiline{
    margin: 30px auto;
    width: 30%;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
#numbercount{
    margin-left: 170px;
}

```

Style appearance in EJ2 JavaScript Textbox control

The [Theme Studio](#) for Essential JS 2 can be used to customize a new theme from an existing theme. It allows modification and preview for primary font and color as well as secondary font and color of the chosen theme. This section covers the instructions and techniques to customize the appearance and styling of the TextBox component beyond the basic requirement provided by the theme studio.

Note: The styling and customization provided in this section will be applied globally for all the Textbox in the page when used as such. To apply the customization only to a specific textbox component, the [cssClass](#) property can be used. [cssClass](#) API adds the custom class attribute to the textbox component, using which the styling can be restricted to the specific component.

INDEX.TS

```

import { TextBox } from '@syncfusion/ej2-inputs';
// Initialize TextBox component
let textbox: TextBox = new TextBox({
    placeholder: 'First Name',
    floatLabelType: 'Auto',
});
textbox.appendTo('#firstname');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>Essential JS 2 TextBox</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Essential JS 2 TextBox Components">
    <meta name="author" content="Syncfusion">
    <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <div id="container">
        <div class="wrap">
            <div id="input-container">
                <input id="firstName">
            </div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 30px 10px;
    width: 260px;
}
.e-input-group .e-style {
    background-color: antiquewhite;
}

```

Filled and Outline mode

The Filled and Outline modes can be enabled in the Textbox component by adding the `e-outline` or `e-filled` class to the [cssClass](#) API.

INDEX.TS

```
import { TextBox } from '@syncfusion/ej2-inputs';
// Initialize TextBox component with cssClass 'e-filled'
let filledTextBox: TextBox = new TextBox({
  placeholder: 'Filled',
  cssClass: 'e-filled',
  floatLabelType: 'Auto'
});
filledTextBox.appendTo('#filled');
// Initialize TextBox component with cssClass 'e-outline'
let outlinedTextBox: TextBox = new TextBox({
  placeholder: 'Outlined',
  cssClass: 'e-outline',
  floatLabelType: 'Auto'
});
outlinedTextBox.appendTo('#outlined');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>Essential JS 2 Multiline</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Essential JS 2 TextBox Component">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <label class="custom-input-label"> Filled TextBox </label>
  <div id="container">
    <input id="filled">
  </div>
  <label class="custom-input-label"> Outlined TextBox </label>
  <div id="container1">
    <input id="outlined">
  </div>
```



```

<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.multiline{
    margin: 30px auto;
    width: 30%;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Note: Filled and Outline theme customization are available only with Material theme.

How To

Set the rounded corner in EJ2 JavaScript Textbox control

Render the TextBox with **rounded corner** by adding the **e-corner** class to the input parent element.

This rounded corner visible only in box model input component

INDEX.TS

```

// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
    inputElement[i].addEventListener("focus", function () {
        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.add('e-input-focus');
        } else {
            this.parentNode.classList.add('e-input-focus');
        }
    });
    inputElement[i].addEventListener("blur", function () {
        let parentElement = this.parentNode;
        if (parentElement.classList.contains('e-input-in-wrap')) {
            parentElement.parentNode.classList.remove('e-input-focus');
        } else {

```

```

        this.parentNode.classList.remove('e-input-focus');
    });
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (let i = 0; i < inputIcon.length; i++) {
    inputIcon[i].addEventListener('mousedown', function () {
        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        let element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Essential JS 2 TextBox</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 TextBox Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/bootstrap.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/bootstrap.css" rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <script src="systemjs.config.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <div class='wrap'>
            <div id="input-container">
                <div class="e-input-group e-corner">
                    <input class="e-input" type="text" placeholder="Enter Date" />
                    <span class="e-input-group-icon e-input-popup-date"></span>
                </div>
                <div class="e-float-input e-input-group e-corner">
                    <input type='text' required />
                    <span class="e-float-line"> </span>
                    <label class="e-float-text">Enter Date </label>
                    <span class="e-input-group-icon e-input-popup-date"></span>
                </div>
            </div>
        </div>
    </div>

```

```

        </div>
    </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

STYLES.CSS

```

.e-corner .e-input-group-icon.e-input-popup-date::before {
    content: '\e960';
}
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 14px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.wrap {
    box-sizing: border-box;
    margin: 0 auto;
    padding: 20px 10px;
    width: 340px;
}
.e-input-group-icon:before {
    font-family: e-icons;
}
.e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow:
adjoining-classes */
    font-size: 16px;
}
.e-input-group.e-small .e-input-group-icon.e-input-popup-date { /* csslint
allow: adjoining-classes */
    font-size: 14px;
}
.e-input-group-icon.e-input-popup-date:before { /* csslint allow: adjoining-
classes */
    content: "□";
}
.e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-classes
*/
    content: '\e85e';
}

```

```
.e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-classes */
  content: "⬇";
}
.e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-classes */
  content: '\e7ba';
}
.e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-classes */
  content: '\e814';
}
.e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-classes */
  content: "📅";
}
.e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-classes */
  content: '\e904';
}
.e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-classes */
  content: '\e913';
}
.e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-classes */
  content: '\e837';
}
.e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-classes */
  content: '\e806';
}
#input-container .e-input-group { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
```

Set the disabled state in EJ2 JavaScript Textbox control

Disable the TextBox by adding the `e-disabled` to the input parent element and set `disabled` attribute to the input element.

INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
  inputElement[i].addEventListener('focus', () => {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.add('e-input-focus');
    } else {
```

```

        this.parentNode.classList.add('e-input-focus');
    }
});
inputElement[i].addEventListener('blur', () => {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
        parentElement.parentNode.classList.remove('e-input-focus');
    } else {
        this.parentNode.classList.remove('e-input-focus');
    }
});
}
}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Essential JS 2 TextBox</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 TextBox Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
    <script src="systemjs.config.js"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <div class='wrap'>
            <div id="input-container">
                <input class="e-input" type="text" placeholder="Enter Name"
disabled/>
                <div class="e-float-input e-disabled">
                    <input type='text' required disabled/>
                    <span class="e-float-line"></span>
                    <label class="e-float-text">Enter Name</label>
                </div>
            </div>
        </div>
    </div>
    <script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
    </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
  font-size: 14px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.wrap {
  box-sizing: border-box;
  margin: 0 auto;
  padding: 20px 10px;
  width: 260px;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
```

Set the read only textbox in EJ2 JavaScript Textbox control

You can make the TextBox as **read-only** by setting the **readonly** attribute to the input element.

INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input,.e-float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
  inputElement[i].addEventListener('focus', () => {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.add('e-input-focus');
    } else {
      this.parentNode.classList.add('e-input-focus');
    }
  });
  inputElement[i].addEventListener('blur', () => {
    let parentElement = this.parentNode;
    if (parentElement.classList.contains('e-input-in-wrap')) {
      parentElement.parentNode.classList.remove('e-input-focus');
    } else {
      this.parentNode.classList.remove('e-input-focus');
    }
  });
}
```

```
}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Essential JS 2 TextBox</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 TextBox Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
  <script src="systemjs.config.js"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <div id='loader'>Loading....</div>
  <div id='container'>
    <div class='wrap'>
      <div id="input-container">
        <input class="e-input" type="text" placeholder="Enter Name"
value="John" readonly/>
        <div class="e-float-input">
          <input type='text' required readonly value="John"/>
          <span class="e-float-line"></span>
          <label class="e-float-text e-label-top">Enter Name</label>
        </div>
      </div>
    </div>
  </div>
<script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

STYLES.CSS

```
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
```

```
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.wrap {
  box-sizing: border-box;
  margin: 0 auto;
  padding: 20px 10px;
  width: 260px;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
  margin: 30px 0;
}
```

Add textbox programmatically in EJ2 JavaScript Textbox control

- Create a TypeScript file and import the **Input** modules

from **ej2-inputs** library as shown below.

``ts`

`import {Input} from '@syncfusion/ej2-inputs';`

```

- Pass the **HTML Input** element as parameter to the **createInput** method.
- You can also add the **icons** on the input by passing **buttons** property value with the class name **e-input-group-icon** as parameter to the **createInput** method.

### INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achieve ripple effect when focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
 this.parentNode.classList.add('e-input-focus')
 });
 inputElement[i].addEventListener("blur", function () {
 this.parentNode.classList.remove('e-input-focus')
 });
}
// Add 'e-input-btn-ripple' class to the icon element for achieve ripple effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (let i = 0; i < inputIcon.length; i++) {
 inputIcon[i].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
}
```



```

 });
 inputIcon[i].addEventListener('mouseup', function () {
 var element = this;
 setTimeout(function () {
 element.classList.remove('e-input-btn-ripple');
 }, 500);
 });
}
import { Input, InputObject } from '@syncfusion/ej2-inputs';
let inputObj: InputObject;
let element: HTMLInputElement = <HTMLInputElement>
document.createElement('input');
document.getElementById('input-container').appendChild(element);
Input.createInput({
 element: element,
 properties:{
 placeholder:'Enter Name'
 }
});
let element1: HTMLInputElement =
<HTMLInputElement>document.createElement('input');
document.getElementById('input-container').appendChild(element1);
Input.createInput({
 element: element1,
 buttons: ['e-input-group-icon e-input-down'],
 properties:{
 placeholder:'Enter Value'
 }
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TextBox</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 TextBox Components">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="wrap">
 <div id="input-container">
 </div>

```

```

 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## STYLES.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.wrap {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 260px;
}
.e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow: adjoining-classes */
 font-size: 16px;
}
.e-input-group-icon:before {
 font-family: e-icons;
}
.e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-classes */
 content: '\e85e';
}
.e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-classes */
 content: "□";
}
.e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-classes */
 content: '\e7ba';
}
.e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-classes */
 content: '\e814';
}

```

```
.e-input-group-icon.e-input-popup-date:before { /* csslint allow: adjoining-classes */
 content: "□";
}
.e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-classes */
 content: "□";
}
.e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-classes */
 content: '\e904';
}
.e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-classes */
 content: '\e913';
}
.e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-classes */
 content: '\e837';
}
.e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-classes */
 content: '\e806';
}
#input-container .e-input-group { /* csslint allow: adjoining-classes */
 margin: 30px 0;
}
```

### Add floating label to textbox programmatically in EJ2 JavaScript Textbox control

The Floating Label TextBox floats label above the TextBox after focusing, or entering a value in the TextBox.

Floating label supports the types of actions as given below.

| Type | Description |
|------|-------------|
|------|-------------|

|      |                                                                                                 |
|------|-------------------------------------------------------------------------------------------------|
| Auto | The floating label will float above the input after focusing, or entering a value in the input. |
|------|-------------------------------------------------------------------------------------------------|

|        |                                                       |
|--------|-------------------------------------------------------|
| Always | The floating label will always float above the input. |
|--------|-------------------------------------------------------|

|       |                                                                                   |
|-------|-----------------------------------------------------------------------------------|
| Never | By default, never float the label in the input when the placeholder is available. |
|-------|-----------------------------------------------------------------------------------|

- Create a TypeScript file and import the `Input` modules from `ej2-inputs` library as shown below.

```
`ts
```

```
import {Input} from '@syncfusion/ej2-inputs';
```

```
,
```

- Pass the `HTML Input` element and `floatLabelType` property as `Auto` to the `createInput` method.
- Set the `placeholder` value to the input element via `element attribute` or pass the parameter to the `createInput` method.

The watermark label will be updated based on the specified placeholder value of the Floating Label TextBox.

- You can add the icons on the input by passing buttons property value with the

class name `e-input-group-icon` as parameter to the `createInput` method.

## INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input,.e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
 this.parentNode.classList.add('e-input-focus')
 });
 inputElement[i].addEventListener("blur", function () {
 this.parentNode.classList.remove('e-input-focus')
 });
}
// To get the all icon elements on the input field.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
for (let i = 0; i < inputIcon.length; i++) {
 inputIcon[i].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[i].addEventListener('mouseup', function () {
 var element = this;
 setTimeout(function () {
 element.classList.remove('e-input-btn-ripple');
 }, 500);
 });
}
import { Input, InputObject } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let inputObj: InputObject;
let element: HTMLInputElement =
<HTMLInputElement>document.createElement('input');
document.getElementById('input-container').appendChild(element);
Input.createInput({
 element: element,
 floatLabelType: "Auto",
 properties:{
 placeholder:'Enter Name'
 }
});
let element2: HTMLInputElement =
<HTMLInputElement>document.createElement('input');
document.getElementById('input-container-01').appendChild(element2);
Input.createInput({
 element: element2,
```

```

 floatLabelType: "Always",
 properties:{
 placeholder:'Enter Name'
 }
 });
 let element3: HTMLInputElement =
 <HTMLInputElement>document.createElement('input');
 document.getElementById('input-container-02').appendChild(element3);
 Input.createInput({
 element: element3,
 floatLabelType: "Never",
 properties:{
 placeholder:'Enter Name'
 }
 });
 // Render float label TextBox with icon
 let element4: HTMLInputElement =
 <HTMLInputElement>document.createElement('input');
 document.getElementById('input-container-03').appendChild(element4);
 Input.createInput({
 element: element4,
 floatLabelType: "Auto",
 buttons: ['e-input-group-icon e-input-down'],
 properties:{
 placeholder:'Enter Value'
 }
 });

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TextBox</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 TextBox Components">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="wrap">
 <div id="input-container">
 <h4>FloatLabelType as Auto </h4>
 </div>
 <div id="input-container-01">

```

```

 <h4> FloatLabelType as Always </h4>
 </div>
 <div id="input-container-02">
 <h4> FloatLabelType as Never </h4>
 </div>
 <div id="input-container-03">
 <h4> Float label input with icons </h4>
 </div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008c00;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.wrap {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 260px;
}
.e-input-group-icon:before {
 font-family: e-icons;
}
.e-input-group .e-input-group-icon.e-input-popup-date { /* csslint allow:
adjoining-classes */
 font-size: 16px;
}
.e-input-group-icon.e-input-popup-date:before { /* csslint allow: adjoining-
classes */
 content: "□";
}
.e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-classes
*/
 content: '\e85e';
}

```

```
.e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-classes */
 content: "▣";
}
.e-input-group-icon.e-input-plus:before { /* csslint allow: adjoining-classes */
 content: '\e7ba';
}
.e-input-group-icon.e-input-minus:before { /* csslint allow: adjoining-classes */
 content: '\e814';
}
.e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-classes */
 content: "▣";
}
.e-input-group-icon.e-input-left:before { /* csslint allow: adjoining-classes */
 content: '\e904';
}
.e-input-group-icon.e-input-right:before { /* csslint allow: adjoining-classes */
 content: '\e913';
}
.e-input-group-icon.e-input-reload:before { /* csslint allow: adjoining-classes */
 content: '\e837';
}
.e-input-group-icon.e-input-search:before { /* csslint allow: adjoining-classes */
 content: '\e806';
}
#input-container-03 .e-input-group { /* csslint allow: adjoining-classes */
 margin: 30px 0;
}
#input-container-03 .e-float-input { /* csslint allow: adjoining-classes */
 margin: 30px 0;
}
```

### Change the floating label color of the textbox in EJ2 JavaScript Textbox control

You can change the floating label color of the TextBox for both **success** and **warning** validation states by applying the following CSS styles.

```
`css
/ For Success state /
.e-float-input.e-success label.e-float-text,
.e-float-input.e-success input:focus ~ label.e-float-text,
.e-float-input.e-success input:valid ~ label.e-float-text {
color: #22b24b;
}
```

*/ For Warning state /*

```
.e-float-input.e-warning label.e-float-text,
.e-float-input.e-warning input:focus ~ label.e-float-text,
.e-float-input.e-warning input:valid ~ label.e-float-text {
color: #ffca1c;
}
,
```

## INDEX.TS

```
// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input,.e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
 let parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener("blur", function () {
 let parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (let i = 0; i < inputIcon.length; i++) {
 inputIcon[i].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[i].addEventListener('mouseup', function () {
 let element = this;
 setTimeout(function () {
 element.classList.remove('e-input-btn-ripple');
 }, 500);
 });
}
}
```

## INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
```



```

<head>
 <title>Essential JS 2 TextBox</title>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <meta name="description" content="Essential JS 2 TextBox Components" />
 <meta name="author" content="Syncfusion" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
 <link href="index.css" rel="stylesheet" />
 <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
 <script src="systemjs.config.js"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <div id='loader'>Loading....</div>
 <div id='container'>
 <div class='wrap'>
 <div id="input-container">
 <div class="e-input-group e-float-input e-success">
 <input type="text" required>

 <label class="e-float-text">Success</label>
 </div>
 <div class="e-input-group e-float-input e-warning">
 <input type="text" required>

 <label class="e-float-text">Warning</label>
 </div>
 </div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

### **STYLES.CSS**

```

#container {
 visibility: hidden;
}

#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
}

```

```

 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

.wrap {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 340px;
}

.e-float-input.e-success label.e-float-text{ /* csslint allow: adjoining-classes */
 color: #22b24b;
}
.e-float-input.e-success input:focus ~ label.e-float-text{ /* csslint allow: adjoining-classes */
 color: #22b24b;
}
.e-float-input.e-success input:valid ~ label.e-float-text { /* csslint allow: adjoining-classes */
 color: #22b24b;
}

.e-float-input.e-warning label.e-float-text{ /* csslint allow: adjoining-classes */
 color: #ffcalc;
}
.e-float-input.e-warning input:focus ~ label.e-float-text{ /* csslint allow: adjoining-classes */
 color: #ffcalc;
}
.e-float-input.e-warning input:valid ~ label.e-float-text { /* csslint allow: adjoining-classes */
 color: #ffcalc;
}
.content-wrapper {
 width: 90%;
 margin: 0 auto;
 min-width: 185px;
}
.content-wrapper .row {
 padding: 15px 0;
}

```

Change the color of the textbox based on its value in EJ2 JavaScript Textbox control

You can change the color of the TextBox by validating its value using regular expression in the **keyup** event for predicting the numeric values as demonstrated in the following code sample.

#### INDEX.TS

```
// To get the all input fields and its container.
```

```
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
 let parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener("blur", function () {
 let parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
}
// To Configure RegExp for predicting Numeric values on keyUp event function
function onKeyUp(e) {
 let str = e.value.match(/^[0-9]+$/);
 if (!(str && str.length > 0) && e.value.length) {
 e.parentElement.classList.add('e-error');
 } else {
 e.parentElement.classList.remove('e-error');
 }
}
```

## INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Essential JS 2 TextBox</title>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <meta name="description" content="Essential JS 2 TextBox Components" />
 <meta name="author" content="Syncfusion" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
buttons/styles/material.css" rel="stylesheet" />
 <link href="index.css" rel="stylesheet" />
 <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
 <script src="systemjs.config.js"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>
 <div id='loader'>Loading....</div>
 <div id='container'>
 <div class='wrap'>
 <label>Normal Input</label>
 <div class="e-input-group">
 <input class="e-input" type="text" placeholder="Enter
numeric values" onkeyup="onKeyup(this)" />
 </div>
 <label> Floating Input </label>
 <div class="e-float-input">
 <input type="text" onkeyup="onKeyup(this)" required />

 <label class="e-float-text">Enter numeric values</label>
 </div>
 <div id="input-container">
 </div>
 </div>
 </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## STYLES.CSS

```
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.wrap {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 340px;
}
#input-container .e-float-input { /* csslint allow: adjoining-classes */
 margin: 30px 0;
}
.e-input-group.e-error .e-input { /* csslint allow: adjoining-classes */
 color: #f44336;
}
.e-float-input.e-error input { /* csslint allow: adjoining-classes */
```

```

 color: #f44336;
}
.wrap label { /* csslint allow: adjoining-classes */
 font-weight:bold;
}

```

### Add floating label to read only textbox in EJ2 JavaScript Textbox control

You can achieve floating label for read-only textboxes by adding/removing **e-label-top** and **e-label-bottom** classes to the label element

In this sample, click the update value button to fill the read-only textbox with value and float a label.

#### INDEX.TS

```

function checkFloatingLabel(id) {
 let inputElement= document.getElementById(id);
 let labelElement = inputElement.parentElement.querySelector('.e-float-
text');
 if (inputElement.value !== '') {
 labelElement.classList.remove('e-label-bottom');
 labelElement.classList.add('e-label-top');
 } else {
 labelElement.classList.remove('e-label-top');
 labelElement.classList.add('e-label-bottom');
 }
}
let inputField = document.getElementById('myText');
document.getElementById('valuebtn').onclick = () => {
 (document.getElementsByClassName('myField')[0]).value = '10';
 checkFloatingLabel('myText')
}
document.getElementById('removebtn').addEventListener('click', function()
{
 (document.getElementsByClassName('myField')[0]).value = '';
 checkFloatingLabel('myText')
}))

```

#### INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
 <title>Essential JS 2 TextBox</title>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <meta name="description" content="Essential JS 2 TextBox Components" />
 <meta name="author" content="Syncfusion" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
buttons/styles/material.css" rel="stylesheet" />
 <link href="index.css" rel="stylesheet" />

```

```

<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
<script src="systemjs.config.js"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<div id='loader'>Loading....</div>
<div id='container' class="text_value">
<div id="input-container"></div>
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<div class="e-float-input">
<input class="e-input myField" id="myText"
name="readonlyAttr" type="text" readOnly>

<label class="e-float-text">Enter value</label>
</div>
</div>
</div>
<div class="row">
<div class="col-xs-10 col-sm-10 col-lg-10 col-md-10">
<button class="e-btn update_value" id='valuebtn' >Set
value</button>
<button class="e-btn remove_value" id='removebtn' >Remove
value</button>
</div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Customize the textbox background color and text color in EJ2 JavaScript Textbox control  
You can customize the textbox styles such as background-color, text-color and border-color by overriding its default styles.

To change the styles of the floating label, you must override the style to the input element.

### INDEX.TS

```

// To get the all input fields and its container.
let inputElement = document.querySelectorAll('.e-input-group .e-input, .e-
float-input.e-input-group input');
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (let i = 0; i < inputElement.length; i++) {
inputElement[i].addEventListener("focus", function () {
this.parentNode.classList.add('e-input-focus');
});
}

```

```
inputElement[i].addEventListener("blur", function () {
 this.parentNode.classList.remove('e-input-focus');
});
}
```

## INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Essential JS 2 TextBox</title>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <meta name="description" content="Essential JS 2 TextBox Components" />
 <meta name="author" content="Syncfusion" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
base/styles/material.css" rel="stylesheet" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
inputs/styles/material.css" rel="stylesheet" />
 <link href="//cdn.syncfusion.com/ej2/21.2.3/ej2-
buttons/styles/material.css" rel="stylesheet" />
 <link href="index.css" rel="stylesheet" />
 <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
 <script src="systemjs.config.js"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <div id='loader'>Loading....</div>
 <div id='container'>
 <div class='wrap'>
 <label>Normal Input</label>
 <div class="e-input-group">
 <input class="e-input" type="text" placeholder="First Name"
/>
 </div>
 <label> Floating Input </label>
 <div class="e-float-input">
 <input type="text" required />

 <label class="e-float-text">Last Name</label>
 </div>
 <div id="input-container">
 </div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## STYLES.CSS

```
.wrap {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 260px;
}
.wrap label { /* csslint allow: adjoining-classes */
 font-weight:bold;
}
.wrap .e-float-input { /* csslint allow: adjoining-classes */
 margin:30px 0;
}
.wrap .e-input-group { /* csslint allow: adjoining-classes */
 margin:25px 0;
}
/* To change the background-color and text-color for textbox */
.e-input-group,
.e-float-input,
.e-float-input.e-input-group { /* csslint allow: adjoining-classes */
 background : lightgray;
 color: green;
}
/* To change the border-color of the textbox */
.e-input-group:not(.e-success):not(.e-warning):not(.e-error):not(.e-float-
icon-left),
.e-input-group:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
float-icon-left) { /* csslint allow: adjoining-classes */
 border-color: #0800ff;
}
/* To change the border-color of the floating-label textbox */
.e-float-input input,
.e-float-input:hover:not(.e-input-group):not(.e-success):not(.e-
warning):not(.e-error):not(.e-disabled) input:not([disabled]) { /* csslint
allow: adjoining-classes */
 border-color: #0800ff;
}
```

## Migration from css textbox to javascript textbox in EJ2 JavaScript Textbox control

From v16.3.21 version, the textbox is provided as JavaScript control to achieve the floating label textbox with minimal code. You can find the available textbox properties, methods, and events in the [API reference](#).

The following table describes the migration from CSS textbox to JavaScript textbox control.

### Normal textbox

<!-- markdownlint-disable MD038 -->

Rendering mode	CSS textbox	JavaScript textbox control
-----	-----	-----



| Default rendering | `<div class='e-input-group'><br/><input class='e-input' type='text' placeholder='Enter Value' /><br/></div> | <input id='default' /><br/><br/> //To initiate the default textbox control <br/>var inputobj = new ej.inputs.TextBox({<br/>placeholder: 'Enter Value',<br/> floatLabelType: 'Never'<br/>});<br/>inputObj.appendTo('#default'); |`

| Textbox with clear button | `<div class='e-input-group'><br/><input class='e-input' placeholder='Enter Value' required='true'><br/><span class='e-clear-icon'></span><br/></div><br/><br/>Note: You have to write action for clear button. | <input id='clear-input' /><br/><br/> var inputobj = new ej.inputs.TextBox({<br/>placeholder: 'Enter Value',<br/>showClearButton: 'true',<br/> floatLabelType: 'Never'<br/>});<br/>inputObj.appendTo('#clear-input'); |`

| Validation states | `<div class='e-input-group e-error'><br/><input class='e-input' type='text' placeholder='Enter Value' /><br/></div><br/><br/>Note: Textbox control consists of three types of validation rules such as success, warning, and error. | <input id='validation' /><br/><br/> var inputobj = new ej.inputs.TextBox({<br/>placeholder: 'Enter Value',<br/>cssClass: 'e-error',<br/> floatLabelType: 'Never'<br/>});<br/>inputObj.appendTo<br/>('#validation'); |`

#### Floating label textbox

<!-- markdownlint-disable MD038 -->

#### | Rendering mode | CSS textbox | JavaScript textbox control |

| ----- | ----- | ----- |

| Default rendering | `<div class='e-float-input'><br/><input type='text' required /><br/><span class='e-float-line'></span><br/><label class='e-float-text'>Enter Value</label><br/></div> | <input id='default' /><br/><br/> //To initiate the default textbox control with floating option <br/>var inputobj = new ej.inputs.TextBox({<br/>placeholder: 'Enter Value',<br/> floatLabelType: 'Auto'<br/>});<br/>inputObj.appendTo('#default'); |`

| Textbox with clear button | `<div class='e-float-input e-input-group'><br/><input type='text' required value= 'Clear Input' /><br/><span class='e-float-line'></span><br/><label class='e-float-text'>Enter Value</label><br/><span class='e-clear-icon'></span><br/></div><br/><br/>Note: You have to write action for clear button. | <input id='clear-input' /><br/><br/> //To initiate the textbox control with clear button and floating option<br/>var inputobj = new ej.inputs.TextBox({<br/>placeholder: 'Enter Value',<br/>showClearButton: 'true',<br/> floatLabelType: 'Auto'<br/>});<br/>inputObj.appendTo('#clear-input'); |`

| Validation states | `<div class='e-float-input e-error'><br/><input type='text' required /><br/><span class='e-float-line'></span><br/><label class='e-float-text'>Enter Value</label><br/></div><br/><br/> Note: Textbox control consists of three types of validation rules such as success, warning, and error. | <input id='validation' /><br/><br/> //To initiate the textbox control with validation and floating option<br/>var inputobj = new ej.inputs.TextBox({<br/>placeholder: 'Enter Value',<br/>cssClass: 'e-error',<br/> floatLabelType: 'Auto'<br/>});<br/>inputObj.appendTo<br/>('#validation'); |`

## TimePicker

### Time range in EJ2 JavaScript Timepicker control

TimePicker provides an option to select a time value within a specified range by using the [min](#) and [max](#) properties. The min value should always be lesser than the max value.

When the min and max properties are configured and the selected time value is out-of-range or invalid, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

The value property depends on the min/max with respect to [strictMode](#) property.

The following example allows you to select a time value within a range of **9:00 AM** to **11:30 AM**.

#### INDEX.TS

```
import { TimePicker } from '@syncfusion/ej2-calendars';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
//creates a timepicker with min and max property
let timeObject: TimePicker = new TimePicker({
 //sets the min value
 min: new Date('3/8/2017 9:00 AM'),
 //sets the max value
 max: new Date('3/8/2017 11:30 AM'),
 //sets the value
 value: new Date('3/8/2017 11:00 AM')
});
timeObject.appendTo('#element');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <!--style reference from the TimePicker component-->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
 <link href="style.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="text" id="element">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If the value of **min** or **max** property is changed through code behind you have to update the **value** property to set within the range.

#### Time Range customization using two TimePicker components

Here, two TimePicker components are used to select the start and end time. The below sample illustrates the appointment time selection scenario with the start and end time option.

Before the start time selection, the end time TimePicker is in disable state. When the start time is selected, then you will be able to select the end time or else, need to select the entire business hours 9:00 to 18:00 from the Business Hours option. Once the options are checked, both the TimePicker components goes to readonly state.

#### INDEX.TS

```

import { TimePicker, ChangeEventArgs } from '@syncfusion/ej2-calendars';
import { CheckBox, ChangeEventArgs as checkboxChange } from
 '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
let value: Date;
let isStartTimeChange: Boolean = true;
let startTime: TimePicker = new TimePicker({
 change: onEnableEndTime
});
startTime.appendTo('#start');
let endTime: TimePicker = new TimePicker({
 enabled: false
});
endTime.appendTo('#end');
let checkboxObject: CheckBox = new CheckBox({ label: 'Business Hours',
change: changeTime });
checkboxObject.appendTo('#dayRange');
let endInput: HTMLInputElement =
<HTMLInputElement>document.getElementById('end');
function changeTime(args: checkboxChange): void {
 /*To determine whether we have selected business hours or not*/
 isStartTimeChange = false;
 if (args.checked) {

```

```

 /*Business hours*/
 startTime.value = new Date('9/6/2017 9:00');
 endTime.enabled = true;
 endTime.value = new Date('9/6/2017 18:00');
 startTime.readonly = true;
 endTime.readonly = true;
 } else {
 endTime.value = null;
 startTime.value = null;
 endInput.value = '';
 startTime.readonly = false;
 endTime.readonly = false;
 endTime.enabled = false;
 }
}
function onEnableEndTime(args: ChangeEventArgs): void {
 /*Enables end time if start time is selected*/
 if (isStartTimeChange) {
 endTime.enabled = true;
 endTime.value = null;
 endInput.value = '';
 value = new Date(+args.value);
 value.setMinutes(value.getMinutes() + endTime.step);
 endTime.min = value;
 } else {
 isStartTimeChange = true;
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <!--style reference from the TimePicker component-->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
 <link href="style.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="pane">
 <div class="tabs-wrap">
 <div class="wrap">
 <input type="text" id="start">
 </div>
 </div>
 <div class="tabs-wrap" style="clear: both">
 <div class="wrap">
 <input type="text" id="end">
 </div>
 </div>
 <div class="tabs-wrap" style="clear: both;padding: 14px 10px;">
 <div class="wrap">
 <input type="checkbox" id="dayRange">
 </div>
 </div>
 </div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Time masking in EJ2 JavaScript Timepicker control

TimePicker has `enableMask` property that provides the option to enable the built-in date masking support. Also, you must inject the MaskedDateTime module to enable the masking support.

### INDEX.TS

```

import { TimePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
TimePicker.Inject(MaskedDateTime);
let timepickerObject: TimePicker = new TimePicker({
 enableMask: true
});
timepickerObject.appendTo('#mask');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 DatePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="styles.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input id="mask">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The mask pattern is defined based on the provided time format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| **Keys | Actions |**

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the time. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion |

The following example demonstrates default and custom format of TimePicker component with mask.

#### INDEX.TS

```

import { TimePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
TimePicker.Inject(MaskedDateTime);
let timepickerObject: TimePicker = new TimePicker({
 enableMask: true
});
timepickerObject.appendTo('#mask');
let timepickerFormat: TimePicker = new TimePicker({
 enableMask: true,
 format: "h:mm a",

```

```
});
timepickerFormat.appendTo('#format');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="styles.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
 <body>

 <div id="container" style="margin:50px auto 0; width:250px;">

 <label class="custom-input-label">Mask support with default
format</label>

 <input id="mask">

 <label class="custom-input-label">Mask support with custom
format</label>

 <input id="format">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

### Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default , it takes the full name of time co-ordinates such as `hour`, `minute` and `second`.

While changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through load method of L10n class for mask placeholder values like below.

```
`ts
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized mask placeholder value
L10n.load({
 'de': {
 'timepicker': { hour: 'Stunde' ,minute: 'Minute', second:'Sekunde' }
 }
});
`
```

The following example demonstrates default and customized mask placeholder value.

#### **INDEX.TS**

```
import { TimePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
TimePicker.Inject(MaskedDateTime);
let timepickerObject: TimePicker = new TimePicker({
 enableMask: true
});
timepickerObject.appendTo('#mask');
let timepickerPlaceholder: TimePicker = new TimePicker({
 enableMask: true,
 maskPlaceholder: {hour: 'h', minute: 'm', second: 's'}
});
timepickerPlaceholder.appendTo('#placeholder');
```

#### **INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="styles.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" style="margin:50px auto 0; width:250px;">

 <label class="custom-input-label">Default mask placeholder</label>

 <input id="mask">

 <label class="custom-input-label">Custom mask placeholder</label>

 <input id="placeholder">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Globalization in EJ2 JavaScript Timepicker control

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [internationalization](#), and also add culture specific customization and translation to the text [localization](#).

By default, the time format and meridian names are specific to the American English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the `loadCldr` method to load culture specific CLDR JSON data. To use a different culture other than English, follow the steps below:

- Install the `CLDR-Data` package by using the following command (installs all the CLDR JSON data). To know more about CLDR-Data refer to the [CLDR-Data](#) link.

```
npm install cldr-data --save
```

Once the package is installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Import the installed CLDR JSON data into the `app.ts` file. To import JSON data, install the JSON plugin loader. Here, The systemJS JSON plugin loader is used.

,

```
npm install systemjs-plugin-json --save-dev
```

,

- After installation, configure the `system.config.js` configuration settings as given below to map the `systemjs-plugin-json` loader.

```
`ts
System.config({
 paths: {
 'npm:': './node_modules/',
 'syncfusion:': 'npm:@syncfusion/'
 },
 map: {
 app: 'app',
 //Syncfusion packages mapping
 "@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
 "@syncfusion/ej2-data": "syncfusion:ej2-data/dist/ej2-data.umd.min.js",
 "@syncfusion/ej2-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
 "@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.umd.min.js",
 "@syncfusion/ej2-buttons": "syncfusion:ej2-buttons/dist/ej2-buttons.umd.min.js",
 "@syncfusion/ej2-lists": "syncfusion:ej2-lists/dist/ej2-lists.umd.min.js",
 "@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.umd.min.js",
 "@syncfusion/ej2-popups": "syncfusion:ej2-popups/dist/ej2-popups.umd.min.js",
 "@syncfusion/ej2-charts": "syncfusion:ej2-charts/dist/ej2-charts.umd.min.js",
 "@syncfusion/ej2-calendars": "syncfusion:ej2-calendars/dist/ej2-calendars.umd.min.js",
 "@syncfusion/ej2-grids": "syncfusion:ej2-grids/dist/ej2-grids.umd.min.js",
 "cldr-data": 'npm:cldr-data',
 "plugin-json": "npm:systemjs-plugin-json/json.js"
 },
```

```

meta: {
 '*.json': { loader: 'plugin-json' }
},
packages: {
 'app': { main: 'app', defaultExtension: 'js' },
 'cldr-data': { main: 'index.js', defaultExtension: 'js' }
}
});
System.import('app');
`

```

- Use the [loadCldr](#) method to load the culture specific CLDR JSON data from the installed location to `app.ts` file.
- TimePicker displayed `Sunday` as the first day of week based on default culture ("en-US"). If you want to display the TimePicker with loaded culture's first day of week, you need to import `weekdata.json` file from the `cldr-data/supplemental` as given in the code example.

```

`ts
import { loadCldr } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
 require('cldr-data/supplemental/numberingSystems.json'),
 require('cldr-data/main/de/ca-gregorian.json'),
 require('cldr-data/main/de/numbers.json'),
 require('cldr-data/supplemental/weekdata.json') // To load the culture based first day of week
);
`

```

- Before changing to a culture other than `English`, ensure that the locale text for the concerned culture is loaded through [L10n.load](#) method.

```

`ts
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized placeholder value
L10n.load({
 'de': {

```

```
'timepicker': { placeholder: 'Wählen Sie Zeit' }
}
});
`
```

- Set the culture by using the [locale](#) property. In the following code example, the TimePicker component is initialized in **German** culture with corresponding localized text.

```
`ts
import { TimePicker } from '@syncfusion/ej2-calendars';
//Load the L10n from ej2-base
import { L10n, loadCldr } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
declare var require: any;
loadCldr(
 require('cldr-data/supplemental/numberingSystems.json'),
 require('cldr-data/main/de/ca-gregorian.json'),
 require('cldr-data/main/de/numbers.json')
);
//load the locale object to set the localized placeholder value
L10n.load({
 'de': {
 'timepicker': { placeholder: 'Wählen Sie Zeit' }
 }
});
// creates the timepicker with German culture.
let timeObject: TimePicker = new TimePicker({
 locale:'de',
 value: new Date()
});
timeObject.appendTo('#element');
```

The following example demonstrates the TimePicker in **German** culture.

**INDEX.TS**

```
import { TimePicker } from '@syncfusion/ej2-calendars';
//Load the L10n from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
//load the CLDR data files.
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
loadCldr(numberingSystems, gregorian, numbers);
L10n.load({
 'de': {
 'timepicker': { placeholder: 'Wählen Sie Zeit' }
 }
});
// creates the timepicker with German culture.
let timeObject: TimePicker = new TimePicker({
 locale: 'de',
 value: new Date()
});
timeObject.appendTo('#element');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <!--style reference from the TimePicker component-->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="style.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input id="element" type="text">
```

```

 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Right-To-Left

The TimePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction.

The code example demonstrates the TimePicker component in **Arabic** culture. It also explains how to set localized text to the placeholder using [L10n.load](#) method.

```

`ts
import { TimePicker } from '@syncfusion/ej2-calendars';
//Load the L10n from ej2-base
import { L10n, loadCldr } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
declare var require: any;
loadCldr(
 require('cldr-data/supplemental/numberingSystems.json'),
 require('cldr-data/main/ar/ca-gregorian.json'),
 require('cldr-data/main/ar/numbers.json')
);
//load the locale object to set the localized placeholder value
L10n.load({
 'ar': {
 'timepicker': { placeholder: 'حدد الوقت' }
 }
});
// creates the timepicker with Arabic culture.
let timeObject: TimePicker = new TimePicker({
 //sets the locale
 locale: 'ar',

```

```

value: new Date(),
//sets the enableRtl
enableRtl: true
});
timeObject.appendTo('#element');

```

The following example demonstrates TimePicker in **Arabic** culture with right-to-left direction.

#### INDEX.TS

```

import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { TimePicker } from '@syncfusion/ej2-calendars';
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
loadCldr(numberingSystems, gregorian, numbers);
//set the placeholder value
L10n.load({
 'ar': {
 'timepicker': { placeholder: 'حدد الوقت' }
 }
});
// creates the timepicker with Arabic culture.
let timeObject: TimePicker = new TimePicker({
 //sets the locale
 locale: 'ar',
 value: new Date(),
 //sets the enableRtl
 enableRtl: true
});
timeObject.appendTo('#element');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <!--style reference from the TimePicker component-->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="style.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input id="element" type="text">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Strict mode in EJ2 JavaScript Timepicker control

The [strictMode](#) is an act that allows you to enter only valid time value within the specified min/max range in the textbox. If the time value is invalid, the component value sets to the previous value. If the time value is out of range, the component sets the time value to min/max value.

The following example demonstrates the TimePicker in `strictMode` with min/max range of **10:00 AM** to **4:00 PM**. It allows you to enter

only valid time within the specified range. If you enter the out-of-range value like **8:00 PM**, the value sets to the max time **4:00 PM** as the value **8:00 PM** is greater than max value of **4:00 PM**. If you enter invalid time value like **9:00 tt**, the value sets to the previous value.

#### INDEX.TS

```

import { TimePicker } from '@syncfusion/ej2-calendars';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
let timeObject: TimePicker = new TimePicker({
 strictMode: true,
 min: new Date('7/18/2017 10:00 AM'),
 max: new Date('7/18/2017 4:00 PM'),
 value: new Date('7/18/2017 8:00 PM'),
});
timeObject.appendTo('#element');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>

```



```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<!--style reference from the TimePicker component-->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="style.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="text" id="element">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the TimePicker act in strictMode **false** state, that allows to enter the invalid or out-of-range time in textbox.

If the time is out-of-range or invalid, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

The following example demonstrates the **strictMode** as **false**. Here, it allows to enter the valid or invalid value in textbox. If you are entering the out-of-range or invalid time value, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

#### INDEX.TS

```

import { TimePicker } from '@syncfusion/ej2-calendars';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
let timeObject: TimePicker = new TimePicker({

```

```

strictMode: false,
min: new Date('7/18/2017 10:00 AM'),
max: new Date('7/18/2017 4:00 PM'),
value: new Date('7/18/2017 8:00 PM'),
});
timeObject.appendTo('#element');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <!--style reference from the TimePicker component-->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="style.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="text" id="element">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If the value of `min` or `max` property is changed through code behind, update the `value` property to set within the range.

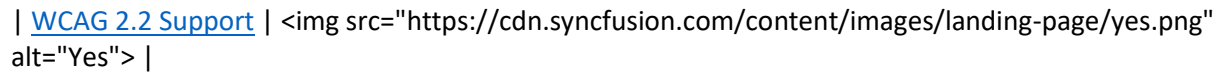
## Accessibility in EJ2 JavaScript Timepicker control

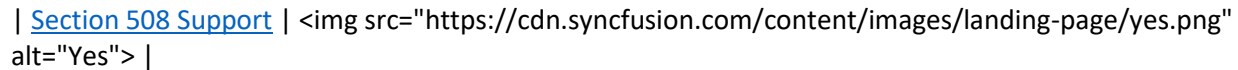
The TimePicker component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

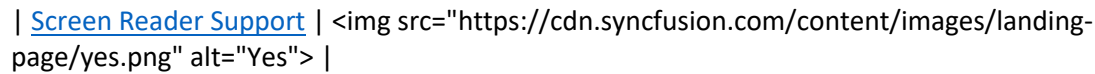
The accessibility compliance for the TimePicker component is outlined below.

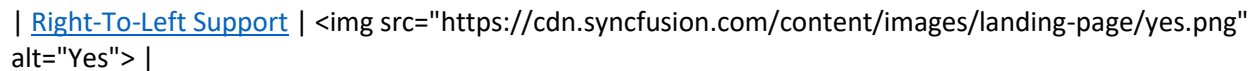
| [Accessibility Criteria](#) | [Compatibility](#) |

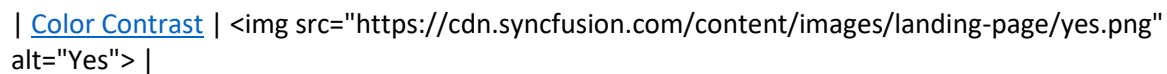
| -- | -- |

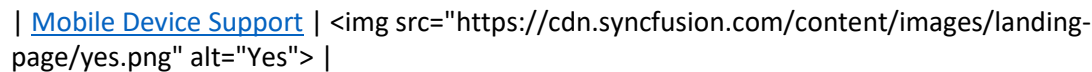
| [WCAG 2.2 Support](#) |  |

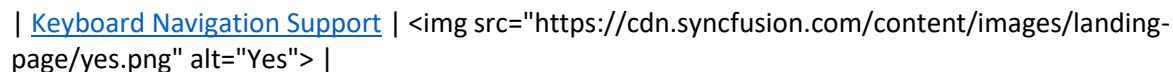
| [Section 508 Support](#) |  |

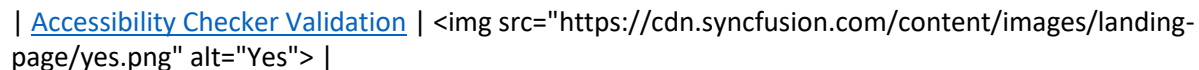
| [Screen Reader Support](#) |  |

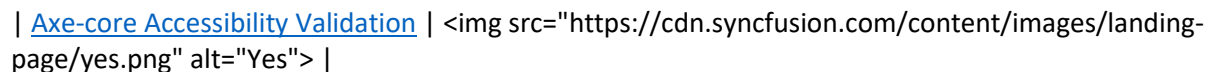
| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

#### WAI-ARIA attributes

The web accessibility makes web applications and its content more accessible to people with disabilities without any barriers. It especially

it tracks the dynamic value changes and DOM changes.

The TimePicker component has covered the [WAI-ARIA](#) specifications with the following list of WAI-ARIA attributes: `aria-haspopup`, `aria-selected`, `aria-disabled`, `aria-activedescendant`, `aria-expanded`, `aria-owns`, and `aria-autocomplete`.

Here in TimePicker, the `combobox` plays the role of input element, and the `listbox` plays the role of popup element.

- **Aria-haspopup:** Provides the information about whether this element display a pop-up window or not.
- **Aria-selected:** Indicates the current selected value of the TimePicker component.
- **Aria-disabled:** Indicates disabled state of the TimePicker component.
- **Aria-expanded:** Indicates the expanded state of the popup.
- **Aria-autocomplete:** Indicates whether user input completion suggestions are provided or not.
- **Aria-owns:** Attribute that creates a parent/child relationship between two DOM element in the accessibility layer.
- **Aria-activedescendent:** Attribute that helps in managing the current active child of the TimePicker component.
- **Role:** Attribute that gives assistive technology information for handling each element in a widget.

### Keyboard Interaction

Keyboard accessibility is one of the most important aspects of web accessibility. Disabled people like blind and those who have motor disabilities or birth defects use keyboard shortcuts more than the mouse.

The TimePicker component has built-in keyboard accessibility support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys to interact with the TimePicker control.

| Press | To do this |

| --- | --- |

| Upper Arrow | Navigate and select the previous item. |

| Down Arrow | Navigate and select the next item. |

| Left Arrow | Move the cursor towards arrow key pressed direction. |

| Right Arrow | Move the cursor towards arrow key pressed direction. |

| Home | Navigate and select the first item. |

| End | Navigate and select the last item. |

| Enter | Select the currently focused item and close the popup. |

| Alt + Upper Arrow | Close the popup. |

| Alt + Down Arrow | Open the popup. |

| Esc | Close the popup. |

In the below sample use the `alt+t` keys to focus the TimePicker component.

**INDEX.TS**

```
import { TimePicker } from '@syncfusion/ej2-calendars';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
// creates the simple timepicker component
let timeObject: TimePicker = new TimePicker({
 placeholder: 'Select Time'
});
timeObject.appendTo('#element');
document.onkeyup = function (e) {
 if (e.altKey && e.keyCode === 84 /* t */) {
 // press alt+t to focus the component by calling public method.
 timeObject.focusIn();
 }
};
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <!--style reference from the TimePicker component-->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">
 <link href="style.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input id="element" type="text">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Ensuring accessibility

The TimePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the TimePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the TimePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

### Style appearance in EJ2 JavaScript Timepicker control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the appearance of TimePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

,

*/ To specify height and font size /*

```
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {
```

```
font-size: 20px;
```

```
height: 40px;
```

```
}
```

,

#### Customizing the TimePicker icon element

Use the following CSS to customize the TimePicker icon element

,

*/ To specify background color and font size /*

```
.e-time-wrapper .e-time-icon.e-icons, *.e-control-wrapper.e-time-wrapper .e-time-icon.e-icons {
```

```
font-size: 20px;
```

```
background-color: beige;
```

```
}
```

,

#### Customizing the TimePicker popup

Use the following CSS to customize the TimePicker popup

,

*/ To specify height /*

```
.e-timepicker.e-popup {
height: 100px;
}
`
```

### Customizing the TimePicker popup content

Use the following CSS to customize the TimePicker popup content

*/ To specify height /*

```
.e-timepicker.e-popup .e-list-parent.e-ul li.e-list-item {
background-color: beige;
font-size: 20px;
}
`
```

### Full screen mode support in mobiles and tablets

The TimePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the TimePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the popup element to occupy the entire screen on mobile devices.

```
`typescript
import { TimePicker } from '@syncfusion/ej2-calendars';
// creates a timepicker with fullScreenMode property
let timeObject: TimePicker = new TimePicker({
// Enable Full Screen Mode
fullScreenMode: true,
});
timeObject.appendTo('#element');
`
```

**Default Sample**

12:00 AM







## How To

### Css customization in EJ2 JavaScript Timepicker control

TimePicker allows you to customize the textbox and popup list appearance to suit your application by using [cssClass](#) property.

The below sample demonstrates customization of text appearance in a textbox, popup button, and popup list along with hover and active

state by using `e-custom-style` class. Following is the list of available classes used to customize the entire TimePicker component.

Class Name	Description
---	---
e-time-wrapper	Applied to TimePicker wrapper element.
e-timepicker	Applied to input element and TimePicker popup element.
e-time-wrapper.e-timepicker	Applied to input element only.
e-input-group-icon.e-time-icon	Applied to popup button.
e-float-text	Applied to floating label text element.
e-popup	Applied to popup element.
e-timepicker.e-popup	Applied to TimePicker popup element only.
e-list-parent	Applied to popup UL element.
e-timepicker.e-list-parent	Applied to TimePicker popup UL element only.
e-list-item	Applied to LI elements.
e-hover	Applied to LI element hovering time.
e-active	Applied to active LI element.

### INDEX.TS

```
import { TimePicker } from '@syncfusion/ej2-calendars';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
// creates timepicker
let timeObject: TimePicker = new TimePicker({
 placeholder: 'Select Time',
 // define the custom class
 cssClass: 'e-custom-style'
});
timeObject.appendTo('#element');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
```

```

<meta name="author" content="Syncfusion">
<!--style reference from the TimePicker component-->
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="style.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="text" id="element">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Client side validation using form validator in EJ2 JavaScript Timepicker control

To achieve client side validation in a TimePicker component, use [Essential JavaScript 2 FormValidator](#). It provides an option to customize feedback error messages to the corresponding fields for taking action and resolving the issue.

In the following example, the required field validation is implemented by mapping the name attribute value to the rules property. It validates the TimePicker component and displays the validation message when the textbox value is empty during form post back or focus out.

#### INDEX.TS

```

import { TimePicker } from '@syncfusion/ej2-calendars';
//import the form validator related functions
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
let timeObject: TimePicker = new TimePicker({
 placeholder: 'Select Time'
});
timeObject.appendTo('#element');
let options: FormValidatorModel = {

```

```

rules: {
 //must specify the name attribute value in rules section
 'timevalue': { required: true }
},
customPlacement: (inputElement: HTMLElement, errorElement: HTMLElement)
=> {
 //to place the error message in custom position
 //inputElement - target element where the error text will be
 appended
 //errorElement - error text which will be displayed.
 inputElement.parentElement.appendChild(errorElement);
}
};
let formObject: FormValidator = new FormValidator('#form-element', options);

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 TimePicker control</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <!--style reference from the TimePicker component-->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="style.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <form id="form-element" class="form-vertical">
 <div class="form-group">
 <div class="col-sm-3 control-label">Required</div>
 <div class="col-sm-6">
 <input type="text" id="element" name="timevalue"
class="form-control">
 </div>
 </div>
 </form>
 </div>
</script>

```

```
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## Ej1 api migration in EJ2 JavaScript Timepicker control

This article describes the API migration process of TimePicker component from Essential JS 1 to Essential JS 2.

### Time Selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	Property: value valuejavascript\$("#timepicker").ejTimePicker({value: new Date("5/5/2014 12:00 PM")});	Property: valuejavascriptvar timepicker = new ej.calendars.TimePicker({ value: new Date("05/11/2017 2:00 PM")});timepicker.appendTo('#element');

### Time Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Display time format	Property: timeFormat timeFormatjavascript\$("#timepicker").ejTimePicker({timeFormat: 'hh:mm:ss tt'});	Property: formatjavascriptvar timepicker = new ej.calendars.TimePicker({ format: 'hh:mm:ss'});timepicker.appendTo('#element');

### Time Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum time	Property: minTimejavascript\$("#timepicker").ejTimePicker({ minTime: '10:00 AM'});	Property: minjavascriptvar timepicker = new ej.calendars.TimePicker({ min: new Date('5/5/2019 3:00 AM')});timepicker.appendTo('#element');
Maximum time	Property: maxTimejavascript\$("#timepicker").ejTimePicker({ maxTime: '10:00 AM'});	Property: Maxjavascriptvar timepicker = new ej.calendars.TimePicker({ max: new Date('5/5/2019 8:00 AM')});timepicker.appendTo('#element');
Set current time	Method: setCurrentTime()javascript\$("#timepicker").ejTimePicker() ;var timeObj = \$("#timepicker").data("ejTimePicker");console.log(timeObj.setCurrentTime());	Can be achieved byjavascriptvar timepicker = new ej.calendars.TimePicker({ value: new Date()});timepicker.appendTo('#element');

### Disabled Time Ranges

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Disable time ranges	Property: disableTimeRangesjavascript\$("#timepicker").ejTimePicker({ disableTimeRanges :[{ startTime: '3:00 AM', endTime: '6:00 AM' }, { startTime: '1:00 PM', endTime: '3:00 PM' }, { startTime: '8:00 PM', endTime: '10:00 PM' } ]});	Can be achieved byjavascriptvar timepicker = new ej.calendars.TimePicker({ itemRender: itemRenderHandler});timepicker.appendTo('#element');function itemRenderHandler(args){ if (args.value.getHours() === 4) { args.isDisabled = true; }}

### Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	Property: <code>CssClassjavascript\$("#timepicker").ejTimePicker({ cssClass : 'gradient-lime'});</code>	Property: <code>CssClassjavascriptvar timepicker = new ej.calendars.TimePicker({ cssClass: 'e-custom-style'});timepicker.appendTo('#element');</code>
Popup list customization	Not Applicable	Event: <code>itemRenderjavascriptvar timepicker = new ej.calendars.TimePicker({ itemRender: itemRenderHandler});timepicker.appendTo('#element');</code> <code>function itemRenderHandler(args) { / code block/}</code>
Show/Hide the popup button	Property: <code>showPopupButtonjavascript\$("#timepicker").ejTimePicker({ showPopupButton : false});</code>	Can be achieved by <code>javascriptvar timepicker = new ej.calendars.TimePicker({ focus: onFocus});timepicker.appendTo('#element');</code> <code>function onFocus(args){ timepicker.show();}.e-control-wrapper .e-input-group-icon.e-time-icon { display: none;}</code>
Enable/Disable the rounded corner	Property: <code>showRoundedCornerjavascript\$("#timepicker").ejTimePicker({ showRoundedCorner : true});</code>	Can be achieved by <code>javascriptvar timepicker = new ej.calendars.TimePicker({ cssClass: 'e-custom-style'});timepicker.appendTo('#element');</code> <code>.e-control-wrapper.e-custom-style.e-time-wrapper.e-input-group { border-radius: 4px;}</code>
Enable/Disable the animation	Property: <code>enableAnimationjavascript\$("#timepicker").ejTimePicker({ enableAnimation : true});</code>	Not Applicable
Interval	Property: <code>intervaljavascript\$("#timepicker").ejTimePicker({ interval : 120});</code>	Property: <code>stepjavascriptvar timepicker = new ej.calendars.TimePicker({ step: 120});timepicker.appendTo('#element');</code>
FocusIn event	Event: <code>focusInjavascript\$("#timepicker").ejTimePicker({ focusIn: function (args) { / code block/}});</code>	Event: <code>focusjavascriptvar timepicker = new ej.calendars.TimePicker({ focus: onFocus});timepicker.appendTo('#element');</code> <code>function onFocus(args){ / code block/}</code>
FocusOut event	Event: <code>focusOutjavascript\$("#timepicker").ejTimePicker({ focusIn: function (args) { / code block/}});</code>	Event: <code>blurjavascriptvar timepicker = new ej.calendars.TimePicker({ blur: onBlur});timepicker.appendTo('#element');</code> <code>function onBlur(args) { / code block/}</code>

FocusIn method	Not Applicable	Method: focusIn()javascriptvar timepicker = new ej.calendars.TimePicker({ value : new Date()});timepicker.appendTo('#element');timepicker.focusIn();
FocusOut method	Not Applicable	Event: focusOutjavascriptvar timepicker = new ej.calendars.TimePicker({ value : new Date()});timepicker.appendTo('#element');timepicker.focusOut();
Prevent popup close	Not Applicable	Event: closejavascriptvar timepicker = new ej.calendars.TimePicker({ close: function (args: PreventableEventArgs) { args.cancel = true; }});timepicker.appendTo('#element');timepicker.show();
Prevent popup open	Not Applicable	Event: openjavascriptvar timepicker = new ej.calendars.TimePicker({ open: function (args) { args.cancel = true; }});timepicker.appendTo('#element');timepicker.show();

### Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the RTL	Property: enableRTLjavascript\$("#timepicker").ejTimePicker({ enableRTL: true});	Property: enableRtljavascriptvar timepicker = new ej.calendars.TimePicker({ enableRtl: true});timepicker.appendTo('#element');

### Persistence

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the persistence	Property: enablePersistencejavascript\$("#timepicker").ejTimePicker({ enablePersistence: true});	Property: enablePersistencejavascriptvar timepicker = new ej.calendars.TimePicker({ enablePersistence: true});timepicker.appendTo('#element');

### Validation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Validation rules	Property: validationRulesjavascript\$("#timepicker").ejTimePicker({ validationRules : {required : true},});	Can be achieved byjavascriptvar timepicker = new ej.calendars.TimePicker({ placeholder: 'Enter time'});timepicker.appendTo('#element');var options = { rules: { 'timevalue': { required: true } }}var formObject: FormValidator = new FormValidator('#form-element', options);
Validation messages	Property: validationMessagesjavascript\$("#timepicker").ejTimePicker({ validationRules : {required : true}, validationMessages : {required: "Required time value" }});	Can be achieved byjavascriptvar timepicker = new ej.calendars.TimePicker({ placeholder: 'Enter date'});timepicker.appendTo('#element');var options = { rules: { 'timevalue': { required: [true, 'Time value required'] } }, customPlacement: function (inputElement, errorElement) { inputElement.parentElement.parentElement.appendChild(errorElement); } }; var formObject = new ej.inputs.FormValidator('#form-element', options);

### Common

<!-- markdownlint-disable MD033 -->



Behavior	API in Essential JS 1	API in Essential JS 2
Width	Property: widthjavascript\$("#timepicker").ejTimePicker({width: 200});	Property: widthjavascriptvar timepicker = new ej.calendars.TimePicker({ width: '200px'});timepicker.appendTo('#ele ment');
Readonly	Property: readOnlyjavascript\$("#timepicker").ejTimePick er({ readOnly: true});	Property: Readonlyjavascriptvar timepicker = new ej.calendars.TimePicker({ readOnly: true, value: new Date()});timepicker.appendTo('#ele ment');
Show/Hide the clear button	Not Applicable	Property: showClearButtonjavascriptvar timepicker = new ej.calendars.TimePicker({ showClearButton: true,});timepicker.appendTo('#elem ent');
Height	Property: Heightjavascript\$("#timepicker").ejTimePicker({ height: "50px"});	Can be achieved byjavascriptvar timepicker = new ej.calendars.TimePicker({ cssClass: 'e-custom- style',});timepicker.appendTo('#ele ment');.e-control-wrapper.e- custom-style.e-time-wrapper.e- input-group { height: 35px;}
Html Attributes	Property: HtmlAttributesjavascript\$("#timepicker").ejTim ePicker({ htmlAttributes : {required:"required"}});	Not Applicable
Watermark text	Property: watermarkTextjavascript\$("#timepicker").ejTim ePicker({ watermarkText : "Enter Time"});	Property: placeholderjavascriptvar timepicker = new ej.calendars.TimePicker({ placeholder: 'Select a Time'});timepicker.appendTo('#elem ent');

Enable the TimePicker	Property: enabledjavascript\$("#timepicker").ejTimePicker({ enabled : true});	Property: enabledjavascriptvar timepicker = new ej.calendars.TimePicker({ enabled: true});timepicker.appendTo('#element');
Disable the TimePicker	Method: disable()javascript\$("#timepicker").ejTimePicker().ejTimePicker().data("ejTimePicker");timeObj.disable();	Property: enabledjavascriptvar timepicker = new ej.calendars.TimePicker({ enabled: false});timepicker.appendTo('#element');
Enable/Disable the textBox editing	Not Applicable	Property: allowEditjavascriptvar timepicker = new ej.calendars.TimePicker({ allowEdit: false});timepicker.appendTo('#element');
zIndex	Not Applicable	Property: zIndexjavascriptvar timepicker = new ej.calendars.TimePicker({ zIndex: 2000});timepicker.appendTo('#element');
Specify the placeholder text behavior	Not Applicable	Property: floatLabelTypejavascriptvar timepicker = new ej.calendars.TimePicker({ floatLabelType: 'Always', placeholder: 'Select a time'});timepicker.appendTo('#element');

### Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	Property: localejavascript\$("#timepicker").ejTimePicker({ locale:"en-US"});	Property: localejavascriptvar timepicker = new ej.calendars.TimePicker({ locale: 'en'});timepicker.appendTo('#element');

### Strict Mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Strict mode	Property: enableStrictModejavascript\$("#timepicker").ejTimePicker({ enableStrictMode: true});	Property: strictModejavascriptvar timepicker = new ej.calendars.TimePicker({ strictMode: true,});timepicker.appendTo('#element');

## Open and Close

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Close	Event: closejavascript\$("#timepicker").ejTimePicker({ close: function(args) { / code block/}});	Event: closejavascriptvar timepicker = new ej.calendars.TimePicker({ close: function (args) { / code block/}});timepicker.appendTo('#element');
Open	Event: openjavascript\$("#timepicker").ejTimePicker({ open: function(args) { / code block/}});	Event: openjavascriptvar timepicker = new ej.calendars.TimePicker({ open: function (args) { / code block/}});timepicker.appendTo('#element');
Hide	Method: hide()javascript\$("#timepicker").ejTimePicker();var timeObj = \$("#timepicker").data("ejTimePicker");timeObj.show();timeObj.hide();	Method: hide()javascriptvar timepicker = new ej.calendars.TimePicker({});timepicker.appendTo('#element');timepicker.show();timepicker.hide();
Show	Method: show()javascript\$("#timepicker").ejTimePicker();var timeObj = \$("#timepicker").data("ejTimePicker");timeObj.show();timeObj.hide();	Method: show()javascriptvar timepicker = new ej.calendars.TimePicker({});timepicker.appendTo('#element');timepicker.show();

## Toast

### Config in EJ2 JavaScript Toast control

This section explains the steps required to customize the appearance of the toast using built-in APIs.

### Title and content template

Toast can be created with the notification message. The message contains [title](#) and [content](#) of the toasts. The title and contents are adaptable in any resolution.

The Title or Content property can be given as HTML element/element ID to a string that can be displayed as a toast.

#### INDEX.TS

```
import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 position: { X: "Center" }
});
toast.appendTo('#element');
toast.show();
document.getElementById('show_toast').onclick = () => {
 toast.show();
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Specifying custom target

By default, the toast can be rendered in the document body. You can change the target position for toast rendering using the [target](#) property. Based on the target, the [position](#) will be updated.

### Close button

By default, the [showCloseButton](#) is not enabled. You can enable it by setting the true value. Before expiring the toast, you can use this button to close or destroy toasts manually.

### Progress bar

By default, the [showProgressBar](#) is not enabled. If it is enabled, it can visually indicate when will the toast gets expired. Based on the [timeOut](#) property, progress bar will appear.

### Progress bar direction

By default, the [progressDirection](#) is set to "Rtl" and it will appear from right to left direction. You can change the progressDirection to "Ltr" to make it appear from left to right direction.

### Newest on top

By default, the newly created toasts will append next with existing toasts. You can change the sequence like inserting before the toast by enabling the [newestOnTop](#).

Here, The following sample demonstrates the combination of the [target](#), [showCloseButton](#), [showProgressBar](#), and [newestOnTop](#) properties in toast.

### INDEX.TS

```

import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'File Downloading',
 content: '<div class="progress"></div>',
 showCloseButton: true,
 target: '#toast_target',
 position: { X: "Center" }
 newestOnTop: true,
 showProgressBar: true,
 progressDirection: "Ltr"
});
toast.appendTo('#element');
toast.show();
document.getElementById('show_toast').onclick = () => {
 toast.show();
}

```

### INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
 <title>Essential JS 2 Toast</title>

```

```

<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Typescript Toolbar Controls" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
 <style>
 #toast_target {
 height: 200px;
 }
 </style>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <div id='loader'>LOADING....</div>
 <div id='container'>
 <div class='row'> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id='element'></div>

 <div id='toast_target'></div>
 </div>
 <script>
 var ele = document.getElementById('container');
 if(ele) {
 ele.style.visibility = "visible";
 }
 </script>
 <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

### Width and height

The dimensions of the toast can be set using the [width](#) and [height](#) properties. This will individually set all toasts. You can create different custom dimension toasts.

By default, the toast can be rendered with **300px** width with **auto** height.

In mobile devices, the default width of the toast gets '100%' width of the page.

When you set toast width as '100%', the toast occupies full width and will be displayed at the top or bottom based on the position **Y** property.

Both the width and height properties allow setting pixels/numbers/percentage. The number value is considered as pixels.

### INDEX.TS

```

import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';

```

```

import { RadioButton, ChangeEventArgs as CheckBoxChange, CheckBox,
ChangeEventArgs } from '@syncfusion/ej2-buttons';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 position: { X: "Center", Y: "Bottom" },
 width: 400,
 height: 150,
});
toast.appendTo('#element');
toast.show();
let timeDelay = 500;
let radioButton: RadioButton = new RadioButton({ label: 'Top', name:
'toast', value: 'Target', change: checkboxChange });
radioButton.appendTo('#topAlign');
let radioButton2 = new RadioButton({ label: 'Bottom', name: 'toast', value:
'Global', checked: true, change: checkboxChange1 });
radioButton2.appendTo('#bottomAlign');
let checkBoxObj: CheckBox = new CheckBox({ label: '100% Width', change:
onChange });
checkBoxObj.appendTo('#fullWidth');
function onChange(e: ChangeEventArgs): void {
 if (e.checked) {
 toast.hide();
 toast.width = "100%";
 toast.title = "";
 toast.content = "<div class='e-custom'>Take a look at our next generation
Javascript <a href='https://ej2.syncfusion.com/home/index.html'
target='_blank'>LEARN MORE</div>";
 toastShow(timeDelay);
 } else {
 toast.hide();
 toast.width = 300;
 toast.title = 'Matt sent you a friend request',
 toast.content = 'You have a new friend request yet to accept',
 toastShow(timeDelay);
 }
}
function checkboxChange(e: CheckBoxChange): void {
 if (e.event.target.checked) {
 toast.position.Y = "Top";
 toast.hide();
 toastShow(timeDelay);
 }
}
function checkboxChange1(e: CheckBoxChange): void {
 if (e.event.target.checked) {
 toast.position.Y = "Bottom";
 toast.hide();
 toastShow(timeDelay);
 }
}
function toastShow(timeOutDelay: number): void {
 setTimeout(
 () => {
 toast.show();

```

```

 }, timeOutDelay);
}
document.getElementById('show_toast').onclick = () => {
 toast.show();
}

```

## INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <meta name="description" content="Typescript Toolbar Controls" />
 <meta name="author" content="Syncfusion" />
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" />
 <link href="index.css" rel="stylesheet" />
 <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <div id='loader'>LOADING....</div>
 <div id='container'>
 <div id='element'></div>
 <div class='row'> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div class='row' style="padding-top: 20px" id=
"toast_pos_target">
 <table>
 <tr>
 <td>
 <div style='padding:25px 0 0 0;'>
 <input id="topAlign" type="radio">
 </div>
 </td>
 <td>
 <div style='padding:25px 0 0 0;'>
 <input id="bottomAlign" type="radio">
 </div>
 </td>
 </tr>
 <tr>
 <td>
 <div style='padding:25px 0 0 0;'>
 <input id="fullWidth" type="checkbox">
 </div>
 </td>
 </tr>
 </table>
 </div>


```



```
<div id='result'></div>
</div>
<script>
 var ele = document.getElementById('container');
 if(ele) {
 ele.style.visibility = "visible";
 }
</script>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
```

### See Also

- [Prevent duplicate toasts](#)
- [Customize the progress bar](#)

### Position in EJ2 JavaScript Toast control

The toast position can be updated based on predefined positions or customizable positions. The predefined position combinations are updated in the [X](#) and [Y](#) position properties.

#### Predefined

##### X Positions

- Left
- Center
- Right

##### Y Positions

- Top
- Bottom

In multiple toast display, the new toast position will not be updated on dynamic change of property values until the old toast messages removed.

The toast occupies full width when you set the width to '100%', so the X positions will not affect the changes when the width is '100%'.

#### Custom

Custom [X](#) and [Y](#) positions can be given as pixels/numbers/percentage. The number value is considered as pixels based on the top and left values updated in the toast.

#### INDEX.TS

```
import { Toast } from '@syncfusion/ej2-notifications';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { RadioButton, ChangeEventArgs as CheckBoxChange } from '@syncfusion/ej2-buttons';
let toastObj: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
```

```

 icon: 'e-laura',
 target: document.body,
 position: { X: 'Right', Y: 'Bottom' }
 });
let listObj: DropDownList = new DropDownList({
 index: 5,
 // set the placeholder to DropDownList input element
 placeholder: 'Select a position',
 // set the height of the popup element
 popupHeight: '200px',
 // bind the change event
 change: valueChange
});
listObj.appendTo('#position');
let radioButton: RadioButton = new RadioButton({ label: 'Target', name:
'toast', value: 'Target', change: checkboxChange });
radioButton.appendTo('#radio1');
let radioButton2 = new RadioButton({ label: 'Global', name: 'toast', value:
'Global', checked: true, change: checkboxChange1 });
radioButton2.appendTo('#radio2');
let radioButton1 = new RadioButton({ label: 'Position', name: 'toastPos',
value: 'Position', checked: true, change: checkboxChange2 });
radioButton1.appendTo('#dropdownRadio');
let radioButton3 = new RadioButton({ label: 'Custom', name: 'toastPos',
value: 'Custom', change: checkboxChange3 });
radioButton3.appendTo('#customRadio');
toastObj.appendTo('#toast_pos');
toastShow(200);
function toastShow(timeOutDelay: number): void {
 setTimeout(
 () => {
 toastObj.show();
 }, timeOutDelay);
}
document.getElementById('show_Toast').onclick = () => {
 if (customFlag) {
 setcustomPosValue();
 }
 toastObj.show();
};
document.getElementById('hideToast').onclick = () => {
 toastObj.hide('All');
};
let customFlag: boolean = false;
function checkboxChange(e: CheckBoxChange): void {
 if (this.checked) {
 toastObj.hide('All');
 toastObj.target = '#toast_pos_target';
 toastShow(1000);
 }
}
function checkboxChange1(e: CheckBoxChange): void {
 if (this.checked) {
 toastObj.hide('All');
 toastObj.target = document.body;
 toastShow(1000);
 }
}

```

```

}
function checkboxChange2(e: CheckBoxChange): void {
 if (this.checked) {
 toastObj.hide('All');
 document.getElementById('dropdownChoose').style.display = 'table-
cell';
 document.getElementById('customChoose').style.display = 'none';
 setToastPosValue(listObj.value.toString()); customFlag = false;
 toastShow(1000);
 }
}
function checkboxChange3(e: CheckBoxChange): void {
 if (this.checked) {
 toastObj.hide('All');
 document.getElementById('dropdownChoose').style.display = 'none';
 document.getElementById('customChoose').style.display = 'table-
cell';
 setcustomPosValue(); customFlag = true; toastShow(1000);
 }
}
//Setting Toast Custom Position
function setcustomPosValue(): void {
 toastObj.position.X =
parseInt((<HTMLInputElement>document.getElementById('xPos')).value, 10);
 toastObj.position.Y =
parseInt((<HTMLInputElement>document.getElementById('yPos')).value, 10);
}
//Setting Toast Position
function setToastPosValue(value: string): void {
 switch (value) {
 case 'topleft':
 toastObj.position.X = 'Left'; toastObj.position.Y = 'Top';
break;
 case 'topright':
 toastObj.position.X = 'Right'; toastObj.position.Y = 'Top';
break;
 case 'topcenter':
 toastObj.position.X = 'Center'; toastObj.position.Y = 'Top';
break;
 case 'topfullwidth':
 toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Top'; break;
 case 'bottomleft':
 toastObj.position.X = 'Left'; toastObj.position.Y = 'Bottom';
break;
 case 'bottomright':
 toastObj.position.X = 'Right'; toastObj.position.Y = 'Bottom';
break;
 case 'bottomcenter':
 toastObj.position.X = 'Center'; toastObj.position.Y = 'Bottom';
break;
 case 'bottomfullwidth':
 toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Bottom'; break;
 }
}
function valueChange(e: ChangeEventArgs): void {

```

```
toastObj.hide('All'); setToastPosValue(e.value.toString());
toastShow(1000);
}
```

## INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <meta name="description" content="Typescript Toolbar Controls" />
 <meta name="author" content="Syncfusion" />
 <link href="index.css" rel="stylesheet" />
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet" />
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>
 <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <div id='loader'>LOADING....</div>
 <div id='container'>
 <div id='element'></div>
 <div class='row' style="padding-top: 20px" id= "toast_pos_target">
 <div id="toast_pos"> </div>
 <div id="toast_pos_property">
 <table style="width: 100%">
 <tr>
 <td>
 <div style='padding:25px 0 0 0;'>
 <input id="dropdownRadio" type="radio">
 </div>
 </td>
 <td>
 <div style='padding:25px 0 0 0;'>
 <input id="customRadio" type="radio">
 </div>
 </td>
 </tr>
 <tr>
 <td id="dropdownChoose" colspan="2">
 <div id="dropdown" style='padding-top:25px;'>
 <select id="position">
 <option value="topleft">Top
Left</option>
 <option value="topright">Top
Right</option>
 <option value="topcenter">Top
Center</option>
```

```

Left</option>
Right</option>
Center</option>
</select>
</div>
</td>
</tr>
<tr>
 | |
```

```

 </div>
 </div>
 <script>
 var ele = document.getElementById('container');
 if(ele) {
 ele.style.visibility = "visible";
 }
 </script>
 <script src="index.js" type="text/javascript"></script>
</body>
</html>

```

#### See Also

- [Render toast with different positions](#)

### Action buttons in EJ2 JavaScript Toast control

You can include action buttons to the toast control by adding the [buttons](#) property. The collection of Essential JS 2 button models can be bound to the `model` property inside the buttons property. You can also include the click event callback function for each button.

#### INDEX.TS

```

import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple, closest } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Anjolie Stokes',
 content: 'Thanks for the update!',
 icon: 'e-laura',
 position: { X: "Right", Y: "Bottom" },
 width: 280,
 height: 120,
 buttons : [{
 model: { content: "Ignore" }, click: btnClick
 }, {
 model: { content: "reply" }
 }]
});
toast.appendTo('#elementToastTime');
toast.show();
document.getElementById('show_toast').onclick = () => {
 toast.show();
}
function btnClick(e: Event) {
 let toastEle = closest(e.target, '.e-toast');
 toast.hide(toastEle);
}

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Toolbar Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<style>
.e-toast-icon.e-laura.e-icons {
 border-radius: 50%;
 background-repeat: no-repeat;
 background-size: cover;
 background-image:
url(https://ej2.syncfusion.com/vue/demos/src/toast/resource/laura.png);
 height: 44px !important;
 margin: 0;
 width: 60px;
}
#elementToastTime .e-toast-message {
 padding: 10px;
}
</style><script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>
 <div id="elementToastTime"></div>

 <div id="result"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [How to add dynamic template](#)

### Timeout in EJ2 JavaScript Toast control

The toast can be expired based on the [timeOut](#) property. The toast can live till the time out reaches without user interaction, a time out value is considered as a millisecond.

- The `timeOut` delay can be visually represented using [Progress Bar](#).
- The [extendedTimeOut](#) property determines how long the toast should be displayed after a user hovers over it.

You can terminate the process by using the [showCloseButton](#) property for destroying the toast at any time.

#### INDEX.TS

```
import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple, closest} from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Anjolie Stokes',
 content: '<p></p>
',
 position: { X: "Right", Y: "Bottom" },
 width: 230,
 height: 250,
 buttons : [{
 model: { content: "Ignore" }
 }, {
 model: { content: "reply" }
 }]
});
toast.appendTo('#elementToastTime');
toast.show();
document.getElementById('show_toast').onclick = () => {
 let value = parseInt(document.getElementById('toast_input_index').value)
 toast.show({timeOut: value});
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
```



```

 <div class="e-float-input"><input class="e-input"
id="toast_input_index" required="" value="0"><span class="e-float-
line"><label class="e-float-text">Enter timeOut</label></div>
 <div class="row"> <button style="margin-top: 20px;" class="e-btn"
id="show_toast"> Show Toast</button> </div>
 <div id="element"></div>
 <div id="elementToastTime"></div>

 <div id="result"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Static toast

You can prevent auto hiding in a toast as visible like static by setting zero (0) value in the timeOut Property.

#### INDEX.TS

```

import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple, closest} from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 showCloseButton: true,
 position: { X: "Right" },
 timeout: 0
});
toast.appendTo('#element');
toast.show();
document.getElementById('show_toast').onclick = () => {
 toast.show();
}

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### See Also

- [Hide the toast on click](#)

### Template in EJ2 JavaScript Toast control

The Template property in toast can be defined as **HTML element**, this can be either a **string** or **selector**.

The HTML element tag can be given as a string for the [Template](#) property.

```
`ts
```

```
template: "<div>Toast Content</div>"
```

```
,
```

The Template property allows getting the template content using the query **selector**. Here, the element 'ID' attribute is specified in the template.

```
`ts
```

```
template: "#Template"
```

```
,
```

### INDEX.TS

```

import { Toast, ToastOpenArgs, ToastCloseArgs, ToastBeforeOpenArgs } from
'@syncfusion/ej2-notifications';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { compile, Browser, closest } from '@syncfusion/ej2-base';
let toast: Toast = new Toast({

```

```

template: document.getElementById('template_toast_ele').innerHTML,
position: { X: 'Right', Y: 'Bottom' },
target: document.body,
extendedTimeout: 0,
timeOut: 120000
});
toast.appendTo('#element');
toast.show();
let btnEle: HTMLElement = document.getElementById('toast_mail_remainder');
let snoozeFlag: boolean = false;
document.getElementById('show_toast').onclick = () => {
 toast.show();
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="template_toast"> <div id="element"></div> </div>

 <div id="result"></div>
 </div>
 <script id="template_toast_ele" type="text/x-template">
 <div id='template_toast' style="display: none">
 <div class="horizontal-align">
 <div class='toast-content'>
 <div class='toast-title'>
 Weekend Alarm
 </div>
 <div class='toast-message'> With traffic, its likely to take
45 minutes to get to jenny's 24th Birthday Bash at Hillside Bar, 454 E.
 Olive Way by 10:00PM </div>
 </div>
 </div>
 </div>

```

```


 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### See Also

- [How to add dynamic template](#)
- [How to play an audio before open the toast](#)

### Animation in EJ2 JavaScript Toast control

The toast control supports custom animations for both shows and hide actions from the provided animation option of the **Animation** library.

The default animation is given as **FadeIn** for showing the toast and **FadeOut** for hiding the toast.

The following sample demonstrates some types of animations that suit toast. You can check all the animation effects here.

### INDEX.TS

```

import {Toast} from '@syncfusion/ej2-notifications';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { enableRipple, Effect } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 position: { X: 'Right', Y: 'Bottom' },
 animation: { show: { effect: "FadeIn" }, hide: { effect: "FadeOut" } },
});
toast.appendTo('#element');
toast.show();
let listObjExpand: DropDownList = new DropDownList({
 index: 0,
 placeholder: 'Select a animate type',
 change: () => { valueChange(); }
});
listObjExpand.appendTo('#showAnimation');
let listObjCollapse: DropDownList = new DropDownList({
 placeholder: 'Select a animate type',
 change: () => { valueChange1(); }
});
listObjCollapse.appendTo('#hideAnimation');
document.getElementById('show_toast').onclick = () => {
 toast.show();
}

```

```

}
function valueChange(): void {
 toast.animation.show.effect = <Effect>(listObjExpand.value);
}
function valueChange1(): void {
 toast.animation.hide.effect = <Effect>(listObjCollapse.value);
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>
 <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="default" style="padding-bottom:75px; width: 35%;">
 <div class="row">
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <label> Show Animation </label>
 </div>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <select id="showAnimation">
 <option value="FadeIn">Fade In</option>
 <option value="FadeZoomIn">Fade Zoom In</option>
 <option value="FadeZoomOut">Fade Zoom Out</option>
 <option value="FlipLeftDownIn">Flip Left Down
In</option>
 <option value="FlipLeftDownOut">Flip Left Down
Out</option>
 <option value="FlipLeftUpIn">Flip Left Up
In</option>
 <option value="FlipLeftUpOut">Flip Left Up
Out</option>
 <option value="FlipRightDownIn">Flip Right Down
In</option>
 <option value="FlipRightDownOut">Flip Right Down
Out</option>
 <option value="FlipRightUpIn">Flip Right Up
In</option>
 </select>
 </div>
 </div>
 </div>
 </div>

```

```

Out</option>
<option value="FlipRightUpOut">Flip Right Up
In</option>
<option value="SlideBottomIn">Slide Bottom
Out</option>
<option value="SlideBottomOut">Slide Bottom
<option value="SlideLeftIn">Slide Left In</option>
<option value="SlideLeftOut">Slide Left Out</option>
<option value="SlideRightIn">Slide Right In</option>
<option value="SlideRightOut">Slide Right
Out</option>
<option value="SlideTopIn">Slide Top In</option>
<option value="SlideTopOut">Slide Top Out</option>
<option value="ZoomIn">Zoom In</option>
<option value="ZoomOut">Zoom Out</option>
</select>
</div>
</div>
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<label> Hide Animation </label>
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<select id="hideAnimation">
<option value="FadeIn">Fade In</option>
<option value="FadeZoomIn">Fade Zoom In</option>
<option value="FadeZoomOut">Fade Zoom Out</option>
<option value="FlipLeftDownIn">Flip Left Down
In</option>
<option value="FlipLeftDownOut">Flip Left Down
Out</option>
<option value="FlipLeftUpIn">Flip Left Up
In</option>
<option value="FlipLeftUpOut">Flip Left Up
Out</option>
<option value="FlipRightDownIn">Flip Right Down
In</option>
<option value="FlipRightDownOut">Flip Right Down
Out</option>
<option value="FlipRightUpIn">Flip Right Up
In</option>
<option value="FlipRightUpOut">Flip Right Up
Out</option>
<option value="SlideBottomIn">Slide Bottom
In</option>
<option value="SlideBottomOut">Slide Bottom
Out</option>
<option value="SlideLeftIn">Slide Left In</option>
<option value="SlideLeftOut">Slide Left Out</option>
<option value="SlideRightIn">Slide Right In</option>
<option value="SlideRightOut">Slide Right
Out</option>
<option value="SlideTopIn">Slide Top In</option>
<option value="SlideTopOut">Slide Top Out</option>
<option value="ZoomIn">Zoom In</option>
<option value="ZoomOut">Zoom Out</option>
</select>

```

```

 </div>
 </div>
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 </div>
 <div id="element"></div>

</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Toast services in EJ2 JavaScript Toast control

The Toast component provides a built-in utility function to render the toast with minimal code. The utility function will render the toast without the need of rendering the container element in the DOM where the toast is appended. So that, the toast can now be rendered on the go. The following are the option to render the toast using the utility function.

#### Show Toast with predefined types

The Toast component support 4 types of predefined toast with essential colors for various situations which can be shown using the `ToastUtility.show` by just defining the type of the toast without defining any class names. The following options are used as an argument on calling the utility function for predefined types:

Options	Description
<a href="#">content</a>	Specifies the content that can be displayed on the Toast.
type	Specifies the type of the predefined Toasts. The 4 types of predefined toasts are <b>Information</b> , <b>Success</b> , <b>Error</b> , <b>Warning</b>
<a href="#">timeOut</a>	Specifies the Toast display time duration on the page in milliseconds. Once the time expires, Toast message will be removed. Setting 0 as a time out value displays the Toast on the page until the user closes it manually.

#### INDEX.TS

```

import { ToastUtility, Toast } from '@syncfusion/ej2-notifications';
let toastObj: Toast;
document.getElementById('info_Toast').onclick = function () {
 toastObj = ToastUtility.show('Please read the comments carefully',
 'Information', 20000);
};
document.getElementById('success_Toast').onclick = function () {
 toastObj = ToastUtility.show('Your message has been sent successfully',
 'Success', 20000);
};

```

```
document.getElementById('error_Toast').onclick = function () {
 toastObj = ToastUtility.show('A problem has been occurred while
submitting the data', 'Error', 20000);
};
document.getElementById('warning_Toast').onclick = function () {
 toastObj = ToastUtility.show('There was a problem with your network
connection', 'Warning', 20000);
};
document.getElementById('hide_Toast').onclick = function () {
 toastObj.hide('All');
};
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div>
 <button class="e-btn e-control e-info" id="info_Toast"> Info
Message </button>
 <button class="e-btn e-control e-success" id="success_Toast">
Success Message </button>
 <button class="e-btn e-control e-warning" id="warning_Toast">
Warning Message </button>
 <button class="e-btn e-control e-danger" id="error_Toast">
Danger Message </button>
 </div>

 <div style="text-align: center;">
 <button class="e-btn e-control" id="hide_Toast"> Hide All
</button>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```



```
</body></html>
```

### Show Toast with ToastModel

The utility function can be called using the [ToastModel](#) as argument to show the toast where all the properties in the [ToastModel](#) like any events, position, close icon, action buttons, etc. can be used in the [ToastUtility.show](#).

#### INDEX.TS

```
import { ToastUtility, Toast } from '@syncfusion/ej2-notifications';
let toastObj: Toast;
document.getElementById('show_Toast').onclick = (): void => {
 toastObj = ToastUtility.show({
 title: 'Toast Title',
 content: 'Toast shown using utility function with ToastModel',
 timeout: 20000,
 position: { X: 'Right', Y: 'Bottom' },
 showCloseButton: true,
 click: toastClick,
 buttons: [{
 model: { content: 'Close' }, click: toastClose
 }]
 });
};
function toastClick() {
 console.log('Toast click event triggered');
}
function toastClose() {
 toastObj.hide();
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div style="text-align: center;">
 <button class="e-btn e-control" id="show_Toast">Show
Toast</button>
```

```

 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Style in EJ2 JavaScript Toast control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the toast title

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

```

`css
/ To change color, font family and font size /
.e-toast-container .e-toast .e-toast-message .e-toast-title {
color: red;
font-size: 18px;
font-weight: bold;
}
`

```

#### Customizing the toast content

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

```

`css
/ To change color, font family and font size /
.e-toast-container .e-toast .e-toast-message .e-toast-content {
color: aqua;
font-size: 13px;
font-weight: normal;
}
`

```

#### Customizing the toast icon

Use the following CSS to customize the default toast icon color.

```

`css

```

*/ To change icon color /*

```
.e-toast-container .e-toast .e-toast-icon {
color: yellow;
}
`
```

### Customizing the toast background

Use the following CSS to customize the default toast's background color.

```
`css
```

*/ To change background color /*

```
.e-toast-container .e-toast {
background-color: navy;
}
`
```

### Accessibility in EJ2 JavaScript Toast control

The toast component has been designed with [WAI-ARIA](#) specifications in mind by applying the prompt WAI-ARIA roles, states, and properties with the keyboard support. It helps users who use assistive WAI-ARIA accessibility support, which is achieved using attributes.

It provides information about the elements in a document for assistive technology.

The toast component implements the keyboard navigation support by using the following [WAI-ARIA practices](#) and is tested in major screen readers.

The Toast component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Toast component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) |  |

| [Section 508 Support](#) |  |

| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

```
| Keyboard Navigation Support | |
| Accessibility Checker Validation | |
| Axe-core Accessibility Validation | |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>
```

#### WAI-ARIA attributes

<!-- markdownlint-disable MD033 -->

| Class | Description |

| ----- | ----- |

| role | <b>alert:</b> Identifies the element as a container when alert content will be added or updated. |

#### INDEX.TS

```
import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 timeout: 0
});
toast.appendTo('#element');
toast.show();
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Toolbar Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Ensuring accessibility

The Toast component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Toast component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Toast component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

### How To

#### Prevent duplicate toast display in EJ2 JavaScript Toast control

You can prevent identical same toast displaying in a screen by the event function and terminate the toast displaying process by setting the cancel event property in the [beforeOpen](#) event.

The following sample demonstrates preventing duplicate title toast element displaying with custom code blocks.

#### **INDEX.TS**

```

import {Toast, ToastModel, ToastBeforeOpenArgs} from '@syncfusion/ej2-
notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toasts: ToastModel[] = [
 { title: 'Warning !', content: 'There was a problem with your network
connection.' , isOpen: false},
 { title: 'Success !', content: 'Your message has been sent
successfully.' , isOpen: false},
 { title: 'Error !', content: 'A problem has been occurred while
submitting your data.' , isOpen: false },
];
let prevDuplicates: boolean = false;
let toastFlag: number = 0;
let toast: Toast = new Toast({
 position: {
 X: 'Center'
 },
 beforeOpen: onBeforeOpen,
 close: onClose,
 created: onCreate
});
toast.appendTo('#element');
++toastFlag;
document.getElementById('show_toast').onclick = () => {
 toast.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {
 toastFlag = 0;
 }
}
function onBeforeOpen(e: ToastBeforeOpenArgs): void {
 if (preventDuplicate(e)) {
 e.cancel = true;
 }
}
function preventDuplicate(e: ToastBeforeOpenArgs): boolean {
 for (let i: number = 0; i < toasts.length; i++) {
 if (toasts[i].title === e.options.title && !toasts[i].isOpen) {
 toasts[i].isOpen = true;
 return false;
 }
 }
 return true;
}
function onCreate(): void {
 toast.show(toasts[toastFlag]);
 ++toastFlag;
}
function onClose(e: any): void {
 for (let i: number = 0; i < toasts.length; i++) {
 if (toasts[i].title === e.options.title) {
 toasts[i].isOpen = false;
 }
 }
}
}

```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

**Restrict the maximum toast to show in EJ2 JavaScript Toast control**

You can restrict the maximum toast count by using the event callback function and terminate the toast displaying process by setting the cancel event property in the [beforeOpen](#) event.

The following sample demonstrates restricting toast displaying up to 3. You can restrict by your own count with custom code blocks.

**INDEX.TS**

```

import {Toast, ToastModel} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let maxCount: number = 3;
let toasts: ToastModel[] = [
 { title: 'Warning !', content: 'There was a problem with your network
connection.' },

```

```

 { title: 'Success !', content: 'Your message has been sent
successfully.' },
 { title: 'Error !', content: 'A problem has been occurred while
submitting your data.' },
 { title: 'Information !', content: 'Please read the comments carefully.'
}
];
let toastFlag: number = 0;
let toast: Toast = new Toast({
 title: 'Sample Toast Title',
 content: 'Sample Toast content'
 position: { X: 'Right', Y: 'Bottom' }
 beforeOpen: onBeforeOpen
});
toast.appendTo('#element');
toast.show(toasts[toastFlag]);
++toastFlag;
document.getElementById('show_toast').onclick = () => {
 toast.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {
 toastFlag = 0;
 }
}
function onBeforeOpen(e: ToastBeforeOpenArgs): void {
 if (maxCount === toast.element.childElementCount) {
 e.cancel = true;
 } else {
 e.cancel = false;
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>
 </div>

```



```


 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Customize progress bar theme and sizing in EJ2 JavaScript Toast control

By default, the progress bar appears based on the theme stylings and dimensions. You can customize the progress bar stylings using custom CSS or event functions.

The following sample demonstrates customizing the progress bar stylings using the [beforeOpen](#) event.

#### INDEX.TS

```

import {Toast, ToastModel} from '@syncfusion/ej2-notifications';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 showProgressBar: true,
 position: { X: 'Right', Y: 'Bottom' }
 beforeOpen: onBeforeOpen
});
let listObjprogressColor: DropDownList = new DropDownList({
 index: 0,
 placeholder: 'Select a animate type',
 popupHeight: '150px',
 change: () => { valueChange(); }
});
listObjprogressColor.appendTo('#Progress');
toast.appendTo('#element');
toast.show();
document.getElementById('show_toast').onclick = () => {
 toast.show();
}
function valueChange() {
 let progressEles: NodeList = toast.element.querySelectorAll('.e-toast-progress');
 progressEles.forEach((ele: HTMLElement)=> {
 ele.style.backgroundColor = listObjprogressColor.value;
 })
}
function onBeforeOpen(e: ToastBeforeOpenArgs): void {
 let progress = e.element.querySelector('.e-toast-progress');
 progress.style.height = document.getElementById('progressHeight').value + 'px';
}

```

```
progress.style.backgroundColor = listObjprogressColor.value;
}
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>
 <div class="toast_row" style="padding-top: 20px">
 <div class="e-float-input">
 <input class="e-input" id="progressHeight" name="Digits"
value="4" required="">

 <label class="e-float-text" for="Digits">Progress Bar
Height</label>
 </div>
 </div>
 <div class="toast_row" style="padding-top: 20px">
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <label> Progress Bar Color </label>
 </div>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <select id="Progress">
 <option value="red">Red</option>
 <option value="cyan">Cyan</option>
 <option value="blue">Blue</option>
 <option value="yellow">Yellow</option>
 <option value="pink">Pink</option>
 </select>
 </div>
 </div>
 <div class="toast_row">
 <button class="e-btn" id="show_toast"> Show Toast</button>
 </div>
 </div>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Play an audio before open the toast in EJ2 JavaScript Toast control

The following sample demonstrates how to play an audio in background while opening the toast by including audio play codes into the beforeOpen event function.

To stop the audio after displaying the toast, use the [open](#) event in toast. For further customization, check the Toast Events [APIs](#).

#### INDEX.TS

```

import {Toast} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 position: { X: 'Right', Y: 'Bottom' }
 beforeOpen: onBeforeOpen
});
toast.appendTo('#element');
document.getElementById('show_toast').onclick = () => {
 toast.show();
}
function onBeforeOpen(e: ToastBeforeOpenArgs): void {
 let audio: HTMLAudioElement = new
Audio('https://drive.google.com/uc?export=download&id=1M95VOptolcQ4FQHzNBaLf
0WFQglrtWi7');
 audio.play();
}

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Show different types of toast in EJ2 JavaScript Toast control

The Essential JS 2 toast has the following predefined styles that can be defined using the [cssClass](#) property for achieving different types of toast:

Class	Description
-----	-----
e-toast-success	Represents a positive toast
e-toast-info	Represents an informative toast
e-toast-warning	Represents a toast with caution
e-toast-danger	Represents a negative toast

### INDEX.TS

```

import {Toast, ToastModel} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toasts: ToastModel[] = [
 { title: 'Warning !', content: 'There was a problem with your network
connection.', cssClass: 'e-toast-warning' },
 { title: 'Success !', content: 'Your message has been sent successfully.',
cssClass: 'e-toast-success'},
 { title: 'Error !', content: 'A problem has been occurred while submitting
your data.', cssClass: 'e-toast-danger' },
 { title: 'Information !', content: 'Please read the comments carefully.',
cssClass: 'e-toast-info' }
];
let toastFlag: number = 0;
let toast: Toast = new Toast({
 position: { X: 'Right', Y: 'Bottom' }
});
toast.appendTo('#element');
toast.show(toasts[toastFlag]);
++toastFlag;
document.getElementById('show_toast').onclick = () => {
 toast.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {

```

```

 toastFlag = 0;
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

### Show multiple toasts in various positions in EJ2 JavaScript Toast control

By default, the positions of the new toasts are only updated after the visible toasts have been destroyed. If You need to display multiple toasts with different positions, initiate another toasts.

The following sample demonstrates adding multiple toasts in different positions.

## INDEX.TS

```

import {Toast, ToastClickEventArgs } from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Warning !',
 content: 'There was a problem with your network connection.'
 position: { X: 'Right', Y: 'Bottom' }
 click: onClick

```

```

});
toast.appendTo('#element');
toast.show();
let toast1: Toast = new Toast({
 title: 'Success !',
 content: 'Your message has been sent successfully.'
 position: { X: 'Left', Y: 'Bottom' }
 click: onClick
});
toast1.appendTo('#element1');
toast1.show();
document.getElementById('show_toast').onclick = () => {
 toast.show();
}
document.getElementById('show_toast1').onclick = () => {
 toast1.show();
}
function onClick(e: ToastClickEventArgs): void {
 e.clickToClose = true;
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast">Show Right
Position Toast</button> </div>
 <div class="row"> <button style="margin-left: 10px;" class="e-btn"
id="show_toast1">Show Left Position Toast</button> </div>
 <div id="element"></div>
 <div id="element1"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

### Close the toast with click tap in EJ2 JavaScript Toast control

By default, the toasts are expired based on the `timeOut` value. You can customize the toast hiding process with click/tap action by setting the event args in the [clicked](#) callback function with [static Toast](#).

#### INDEX.TS

```
import {Toast, ToastClickEventArgs } from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 timeout: 0,
 position: { X: 'Right', Y: 'Bottom' }
 click: onClick
});
toast.appendTo('#element');
toast.show();
document.getElementById('show_toast').onclick = () => {
 toast.show();
}
function onClick(e: ToastClickEventArgs): void {
 e.clickToClose = true;
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
```

```

 </div>
</script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Add dynamic template in EJ2 JavaScript Toast control

Toast supports to change templates dynamically with displaying in multiple toasts. You can change the toast properties while calling in the [show](#) method.

#### INDEX.TS

```

import {Toast, ToastClickEventArgs } from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 position: { X: 'Right', Y: 'Bottom' }
 click: onClick
});
let toastFlag: number = 0;
let toasts = [{ template: '2 Mail has received'},{ template: 'User Guest
Logged in'},{ template: 'Logging in as Guest'},{ template: 'Ticket has
reserved '},{ template: '#templateToast' }];
toast.appendTo('#element');
toast.show(toasts[toastFlag]);
++toastFlag;
document.getElementById('show_toast').onclick = () => {
 toast.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {
 toastFlag = 0;
 }
}
function onClick(e: ToastClickEventArgs): void {
 e.clickToClose = true;
}

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Prevent toast close with mobile swipe in EJ2 JavaScript Toast control

You can prevent the toast close with mobile swipe action by setting [beforeClose](#) argument cancel value to true while argument type as a swipe. The following code shows how to prevent toast close with mobile swipe.

The following sample demonstrates preventing toast close with mobile swipe element displaying with custom code blocks.

#### INDEX.TS

```

import {Toast, ToastBeforeCloseArgs} from '@syncfusion/ej2-notifications';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let toast: Toast = new Toast({
 title: 'Matt sent you a friend request',
 content: 'You have a new friend request yet to accept',
 position: { X: "Center" },
 beforeClose: onBeforeClose,
});
toast.appendTo('#element');
toast.show();
function onBeforeClose (args: ToastBeforeCloseArgs) {
 if (args.type === "swipe") {
 args.cancel = true;
 }
}
document.getElementById('show_toast').onclick = () => {
 toast.show();
}

```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toast</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="row"> <button class="e-btn" id="show_toast"> Show
Toast</button> </div>
 <div id="element"></div>

 <div id="result"></div>
 <div id="templateToast" style="display: none;color:red"> System
affected by virus !!! </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Toolbar

### Getting started in EJ2 JavaScript Toolbar control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

#### Component Initialization

The Essential JS 2 JavaScript components can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

#### *Using local script and style references in a HTML page*

**Step 1:** Create an app folder myapp for Essential JS 2 JavaScript components.

**Step 2:** You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

**Syntax:**

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE\_NAME}.min.js**

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

**Example:**

Script: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-navigations/dist/global/ej2-navigations.min.js**

Styles: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-navigations/styles/material.css**

**Step 3:** Create a folder **myapp/resources** and copy/paste the Toolbar and its dependency scripts and styles from the above installed location to **myapp/resources** location.

**Step 4:** Create a HTML page (index.html) in **myapp** location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Toolbar</title>
<!-- Essential JS 2 Toolbar's dependent material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css" />
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css" />
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css" />
<!-- Essential JS 2 Toolbar's material theme -->
<link href="resources/navigations/styles/material.css" rel="stylesheet" type="text/css" />
<!-- Essential JS 2 Toolbar's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Toolbar's global script -->
```

```
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

**Step 5:** Now, add the **Toolbar** element and initiate the **Essential JS 2 Toolbar** component in the **index.html** by using following code

#### Initialize the Toolbar with commands

The Toolbar can be rendered by defining an array of [items](#). An item is rendered with text by defining the default item type as a **Button**. For more information about item configuration, refer to the [Item Configuration](#) section.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 Toolbar</title>
<!-- Essential JS 2 Toolbar's dependent material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css" />
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css" />
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css" />
<!-- Essential JS 2 Toolbar's material theme -->
<link href="resources/navigations/styles/material.css" rel="stylesheet" type="text/css" />
<!-- Essential JS 2 Toolbar's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Toolbar's global script -->
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element -->
<div id="element"></div>
<script>
```

```
//Initialize Toolbar component
//Initialize Toolbar component
var toolbar = new ej.navigations.Toolbar({
items: [
{ text: 'Cut' },
{ text: 'Copy' },
{ text: 'Paste' },
{ type: 'Separator'},
{ text: 'Bold' },
{ text: 'Italic' },
{ text: 'Underline' },
]
});
//Render initialized Toolbar component
toolbar.appendTo('#element');
</script>
</body>
</html>
,
```

**Step 6:** Now, run the `index.html` in web browser, it will render the **Essential JS 2 Toolbar** component.

*Using CDN link for script and style reference*

**Step 1:** Create an app folder `myapp` for the Essential JS 2 JavaScript components.

**Step 2:** The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

**Syntax:**

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

**Example:**

Script: <http://cdn.syncfusion.com/ej2/ej2-navigations/dist/global/ej2-navigations.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css>

**Step 3:** Create a HTML page (index.html) in myapp location and add the CDN link references. Now, add the **Toolbar** element and initiate the **Essential JS 2 Toolbar** component in the index.html by using following code.

### INDEX.HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>

 <title>Essential JS 2 Toolbar</title>
 <!-- Essential JS 2 Toolbar's dependent material theme -->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css"/>
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet" type="text/css"/>
 <!-- Essential JS 2 Toolbar's material theme -->
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet" type="text/css"/>
 <!-- Essential JS 2 Toolbar's dependent scripts -->
 <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/dist/global/ej2-base.min.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/dist/global/ej2-buttons.min.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/dist/global/ej2-popups.min.js"></script>
 <!-- Essential JS 2 Toolbar's global script -->
 <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
 </head>
 <body>
 <!-- Add the HTML <div> element -->
 <div id="element"></div>
 <script>
//Initialize Toolbar component
var toolbar = new ej.navigations.Toolbar({
items: [
 { text: 'Cut' },
 { text: 'Copy' },
 { text: 'Paste' },
 { type: 'Separator' },
 { text: 'Bold' },
 { text: 'Italic' },
 { text: 'Underline' },
]
});
//Render initialized Toolbar component
toolbar.appendTo('#element');
 </script>
 </body>
</html>
```

**Step 4:** Now, run the `index.html` in web browser, it will render the `Essential JS 2 Toolbar` component.

See Also

- [Initialize the toolbar using HTML elements](#)

### Item configuration in EJ2 JavaScript Toolbar control

The Toolbar can be rendered by defining an array of [items](#). Items can be constructed with the following built-in command types or item template.

#### Button

**Button** is the default command [type](#), and it can be rendered by using the [text](#) property.

Properties of the button command type:

Property | Description

[text](#) | The text to be displayed for button.

[id](#) | The ID of the button to be rendered. If the ID is not given, auto ID is generated.

[prefixIcon](#) | Defines the class used to specify an icon for the button. The icon is positioned before the text if text is available or the icon alone button is rendered.

[suffixIcon](#) | Defines the class used to specify an icon for the button. The icon is positioned after the text if text is available. If both [prefixIcon](#) and [suffixIcon](#) are specified, only [prefixIcon](#) is considered.

[width](#) | Used to set the [width](#) of the button.

[align](#) | Specifies the location for aligning Toolbar items.

#### Separator

The **Separator** type adds a vertical separation between the Toolbar's single/multiple commands.

#### INDEX.TS

```
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 items: [
 { text: "Cut" },
 { text: "Copy" },
 { type: "Separator" },
 { text: "Paste" },
 { type: "Separator" },
 { text: "Undo" },
 { text: "Redo" }
]
});
toolbar.appendTo('#element');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Toolbar Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 <div id="result"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If **Separator** is added as the first or the last item, it will not be visible.

### Input

The **Input** type is only applicable for adding **template** elements when the **template** property is defined as an **object**. Input type creates an **input element** internally that acts as the container for Syncfusion input based components.

Note: Set toolbar item **type** property value as **Input** only for Input components.

### NumericTextBox

- The **NumericTextBox** component can be included by importing the **NumericTextBox** module from **ej2-inputs**.
- Initialize the **NumericTextBox** in **template** property, where the Toolbar item type is set as **Input**.
- Related **NumericTextBox** component properties can also be configured as given below.

```
`javascript
```

```
new NumericTextBox({ format: 'c2' })))
```

```
,
```



### DropDownList

- The **DropDownList** component can be included by importing the **DropDownList** module from **ej2-droppdowns**.
- Initialize the **DropDownList** in template property, where the Toolbar item type is set as **Input**.
- Related **DropDownList** component properties can also be configured as given below.

```
`javascript
new DropDownList({ width:100 })
`
```

### CheckBox

- The **CheckBox** component can be included by importing the **CheckBox** module from **ej2-buttons**.
- Initialize the **CheckBox** in template property, where the Toolbar item type is set as **Input**.
- Related **CheckBox** component properties can also be configured as given below.

```
`javascript
new CheckBox({ label: 'Default' })
`
```

### RadioButton

- The **RadioButton** component can be included by importing the **RadioButton** module from **ej2-buttons**.
- Initialize the **RadioButton** in template property, where the Toolbar item type is set as **Input**.
- Related **RadioButton** component properties can also be configured as given below.

```
`javascript
new RadioButton({ label: 'Option 1', name: 'default'})
`
```

The above steps are applicable for all 'Syncfusion' input based components.

E.g.: The following code explains how to add **NumericTextBox**, **DropDownList**, **RadioButton** and **CheckBox** components to the Toolbar.

### INDEX.TS

```
import { Toolbar } from '@syncfusion/ej2-navigations';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { DropDownList } from '@syncfusion/ej2-droppdowns';
import { CheckBox, RadioButton } from '@syncfusion/ej2-buttons';
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
let toolbar: Toolbar = new Toolbar({
 items: [
```

```

 { text: "Cut" },
 { text: "Copy" },
 { type: "Separator" },
 { text: "Undo" },
 { text: "Redo" },
 { type: "Separator" },
 { type: 'Input', template: new NumericTextBox({ format: 'c2',
value:1, width:150 }) },
 { type: "Separator" },
 { type: 'Input', template: new DropDownList({ dataSource:
sportsData, width: 120, index: 2 }) },
 { type: "Separator" },
 { type: 'Input', template: new CheckBox({ label: 'Checkbox',
checked: true }) },
 { type: "Separator" },
 { type: 'Input', template: new RadioButton({ label: 'Radio', name:
'default', checked: true }) }
]
});
toolbar.appendTo('#element');
```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 <div id="result"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

### *Enabling tab key navigation in Toolbar*

The [tabIndex](#) property of a Toolbar item is used to enable tab key navigation for the item. By default, the user can switch between items using the arrow keys, but the [tabIndex](#) property allows you to switch between items using the Tab and Shift+Tab keys as well.

To use the [tabIndex](#) property, you need to set it for each Toolbar item that you want to enable tab key navigation. The [tabIndex](#) property should be set to a positive integer value. A value of 0 or a negative value will disable tab key navigation for the item.

For example, to enable tab key navigation for two Toolbar items, you can use the following code:

```
`ts
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 items: [
 { text: "Item 1", tabIndex: 1 },
 { text: "Item 2", tabIndex: 2 }
]
});
`
```

With the above code, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, in addition to using the arrow keys. The items will be navigated in the order specified by the [tabIndex](#) values.

If you set the [tabIndex](#) value to 0 for all Toolbar items, tab key navigation will be based on the element order rather than the [tabIndex](#) values. For example:

```
`ts
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 items: [
 { text: "Item 1", tabIndex: 0 },
 { text: "Item 2", tabIndex: 0 }
]
});
`
```

In this case, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated in the order in which they appear in the DOM.

Example:

Here is an example of how you can use the [tabIndex](#) property to enable tab key navigation for a Toolbar component:

#### INDEX.TS

```
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 width: 300,
 overflowMode: 'Scrollable',
 items: [
 { type: 'Button', prefixIcon: 'e-cut-icon tb-icons', text: 'Cut',
tabIndex: 0 },
 { type: 'Button', prefixIcon: 'e-copy-icon tb-icons', text: 'Copy',
tabIndex: 0},
 { type: 'Button', prefixIcon: 'e-paste-icon tb-icons',
text: 'Paste',tabIndex: 0},
 { type: 'Separator'},
 { type: 'Button', prefixIcon: 'e-bold-icon tb-icons', text: 'Bold',
tabIndex: 0},
 { type: 'Button', prefixIcon: 'e-underline-icon tb-icons',
text: 'Underline', tabIndex: 0},
 { type: 'Button', prefixIcon: 'e-italic-icon tb-icons', text: 'Italic',
tabIndex: 0},
 { type: 'Button', prefixIcon: 'e-color-icon tb-icons', text: 'Color-
Picker', tabIndex: 0 },
 { type: 'Separator'},
 { type: 'Button', prefixIcon: 'e-ascending-icon tb-icons', text: 'A-Z
Sort', tabIndex: 0 },
 { type: 'Button', prefixIcon: 'e-descending-icon tb-icons', text: 'Z-A
Sort', tabIndex: 0},
 { type: 'Button', prefixIcon: 'e-clear-icon tb-icons', text: 'Clear',
tabIndex: 0}]
 });
toolbar.appendTo('#element');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>

 <div id="container">
 <div id="element"></div>

 <div id="result"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

With the above code, the user can switch between the Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated based on the element order.

See Also

- [How to set item wise custom template](#)

## Responsive mode in EJ2 JavaScript Toolbar control

This section explains the supported display modes of the Toolbar when the content exceeds the viewing area. Possible modes are:

- Scrollable
- Popup

### Scrollable

The default overflow mode of the Toolbar is **Scrollable**. Scrollable display mode supports display of commands in a single line with horizontal scrolling enabled when commands overflow to available space.

- The right and left navigation arrows are added to the start and end of the Toolbar to navigate to hidden commands.
- You can also see the hidden commands using touch swipe action.
- By default, if navigation icon in the **left** side is disabled, you can see the hidden commands by moving to the **right**.
- By clicking the arrow or by holding the arrow continuously, hidden commands will become visible.
- If device navigation icons are not available, you can touch swipe to see the hidden commands of the Toolbar.



- Once the Toolbar reaches the last or first command, the corresponding navigation icon will be disabled and you can move to the opposite direction.



- You can continuously scroll the Toolbar content by holding the navigation icon.



## INDEX.TS

```
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 width: 600,
 items: [
 { type: 'Button', prefixIcon: 'e-cut-icon tb-icons', text: 'Cut' },
 { type: 'Button', prefixIcon: 'e-copy-icon tb-icons', text: 'Copy' },
 { type: 'Button', prefixIcon: 'e-paste-icon tb-icons', text: 'Paste' },
 { type: 'Separator' },
 { type: 'Button', prefixIcon: 'e-bold-icon tb-icons', text: 'Bold' },
],
});
```

```

 { type: 'Button', prefixIcon: 'e-underline-icon tb-icons',
text:'Underline' },
 { type: 'Button', prefixIcon: 'e-italic-icon tb-icons',
text:'Italic' },
 { type: 'Button', prefixIcon: 'e-color-icon tb-icons',
text:'Color-Picker' },
 { type: 'Separator'},
 { type: 'Button', prefixIcon: 'e-ascending-icon tb-icons',
text:'A-Z Sort' },
 { type: 'Button', prefixIcon: 'e-descending-icon tb-icons',
text:'Z-A Sort' },
 { type: 'Button', prefixIcon: 'e-clear-icon tb-icons',
text:'Clear' }
]
});
toolbar.appendTo('#element');
```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 <div id="result"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

### Popup

**Popup** is another type of [overflowMode](#) in which the Toolbar container holds the commands that can be placed in the available space. The rest of the overflowing commands that do not fit within the viewing area moves to the overflow popup container.

The commands placed in the popup can be viewed by opening the popup using the drop down icon given at the end of the Toolbar.



If the popup content overflows the height of the page, then the rest of the commands will be hidden.

### Priority of commands

Default popup priority is set as **none**, and when the commands of the Toolbar overflow, the ones listed last will be moved to the popup.

You can customize the priority of commands to be displayed on the Toolbar and popup by using the [overflow](#) property.

Property | Description

Both | Button text is visible in both **Toolbar** and **Popup**.

Overflow | Button text is only visible in **Popup**.

Toolbar | Button text is only visible on the **Toolbar**.

In the following code sample, text is only visible in the popup container and not in the Toolbar container.

### INDEX.TS

```
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 width: 330,
 overflowMode: 'Popup',
 items: [
 { type: 'Button', prefixIcon: 'e-cut-icon tb-icons', text: 'Cut',
 showTextOn: 'Overflow', overflow: 'Show' },
 { type: 'Button', prefixIcon: 'e-copy-icon tb-icons', text: 'Copy',
 showTextOn: 'Overflow', overflow: 'Show' },
 { type: 'Button', prefixIcon: 'e-paste-icon tb-icons', text: 'Paste',
 showTextOn: 'Overflow', overflow: 'Show' },
 { type: 'Separator' },
]
});
```



```

 { type: 'Button', prefixIcon: 'e-bold-icon tb-icons', text: 'Bold',
 showTextOn: 'Overflow', overflow: 'Show' },
 { type: 'Button', prefixIcon: 'e-underline-icon tb-icons',
 text: 'Underline', showTextOn: 'Overflow', overflow: 'Show' },
 { type: 'Button', prefixIcon: 'e-italic-icon tb-icons', text: 'Italic',
 overflow: 'Show', showTextOn: 'Overflow' },
 { type: 'Button', prefixIcon: 'e-color-icon tb-icons', text: 'Color-
 Picker', overflow: 'Hide', showTextOn: 'Overflow' },
 { type: 'Separator' },
 { type: 'Button', prefixIcon: 'e-ascending-icon tb-icons', text: 'A-Z
 Sort', overflow: 'Show', showTextOn: 'Overflow' },
 { type: 'Button', prefixIcon: 'e-descending-icon tb-icons', text: 'Z-A
 Sort', overflow: 'Show', showTextOn: 'Overflow' },
 { type: 'Button', prefixIcon: 'e-clear-icon tb-icons', text: 'Clear',
 overflow: 'Show', showTextOn: 'Overflow' }]
 });
 toolbar.appendTo('#element');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 <div id="result"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Template configuration in EJ2 JavaScript Toolbar control

The [JavaScript Toolbar](#) can be rendered by item based collection and by HTML elements. To render it based on the given HTML element, use `id` as the `target` property. To render the Toolbar, follow the below structure of the HTML elements:

```
,
<div id='template_toolbar'> --> Root Toolbar Element
<div> --> Toolbar Items Container
<div> --> Toolbar Item Element
</div>
</div>
</div>
,
```

Here, the template ID, `#template_toolbar` is directly appended to the Toolbar.

### INDEX.TS

```
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar();
toolbar.appendTo('#template_toolbar');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="template_toolbar">
 <div>
 <div><button class="e-btn e-tbar-btn">Cut</button> </div>
 <div><button class="e-btn e-tbar-btn">Copy</button> </div>
 <div><button class="e-btn e-tbar-btn">Paste</button> </div>
 <div class="e-separator"> </div>
 <div><button class="e-btn e-tbar-btn">Bold</button> </div>
 <div><button class="e-btn e-tbar-btn">Italic</button> </div>
 </div>
 </div>
 </div>
```

```

 </div>
 </div>

 <div id="result"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Popup customization

**Popup** is one of the supported responsive modes of the Toolbar. The Toolbar commands, popup mode priority and button text mode customizations are achieved in the item based rendering through property declaration. For more information on popup mode, refer [here](#)

The above behavior can also be achieved with template rendering by defining **equivalent class** names instead of property declaration.

Equivalent class names listed below are needed to add the Toolbar items' **div** element.

#### Priority

Class	Description
e-popup-text	Button text is only visible in the <b>Popup</b> .
e-toolbar-text	Button text is only visible on the <b>Toolbar</b> .

### INDEX.TS

```

import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({ width: 300 , overflowMode: 'Popup' });
toolbar.appendTo('#template_toolbar');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="template_toolbar">
 <div>
 <div class="e-overflow-show e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-cut-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">Cut</div></button> </div>
 <div class="e-overflow-show e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-copy-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">Copy</div></button> </div>
 <div class="e-overflow-show e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-paste-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">Paste</div></button> </div>
 <div class="e-separator"> </div>
 <div class="e-overflow-show e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-bold-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">Bold</div></button> </div>
 <div class="e-overflow-hide e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-underline-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">Underline</div></button> </div>
 <div class="e-overflow-show e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-italic-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">Italic</div></button> </div>
 <div class="e-overflow-show e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-ascending-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">A-Z Sort</div></button> </div>
 <div class="e-overflow-show e-popup-text"><button class="e-
 btn e-tbar-btn"><span class="e-descending-icon tb-icons e-icons e-btn-
 icon"><div class="e-tbar-btn-text">Z-A Sort</div></button> </div>
 </div>
 </div>

 <div id="result"></div>
 </div>
</script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Integrate menu component

You can integrate menu component as toolbar item in Toolbar using [template](#) property. Menu can be populated with items as needed.

### INDEX.TS

```

import { Toolbar, Menu, MenuItemModel } from '@syncfusion/ej2-navigations';
let menuTemplate: string = '<ul id="menu">';
let data: MenuItemModel[] = [

```

```

{
 text: 'Appliances',
 items: [
 {
 text: 'Kitchen',
 items: [
 { text: 'Electric Cookers' },
 { text: 'Coffee Makers' },
 { text: 'Blenders' },
],
 },
 {
 text: 'Washing Machine',
 items: [{ text: 'Fully Automatic' }, { text: 'Semi Automatic' }],
 },
 {
 text: 'Air Conditioners',
 items: [
 { text: 'Inverter ACs' },
 { text: 'Split ACs' },
 { text: 'Window ACs' },
],
 },
],
},
{
 text: 'Accessories',
 items: [
 {
 text: 'Mobile',
 items: [
 { text: 'Headphones' },
 { text: 'Memory Cards' },
 { text: 'Power Banks' },
],
 },
 {
 text: 'Computer',
 items: [
 { text: 'Pendrives' },
 { text: 'External Hard Disks' },
 { text: 'Monitors' },
],
 },
],
},
{
 text: 'Fashion',
 items: [
 {
 text: 'Men',
 items: [
 { text: 'Shirts' },
 { text: 'Jackets' },
 { text: 'Track Suits' },
],
 },
],
},

```

```

 {
 text: 'Women',
 items: [{ text: 'Kurtas' }, { text: 'Salwars' }, { text: 'Sarees'
 }],
 },
],
},
{
 text: 'Home & Living',
 items: [
 {
 text: 'Furniture',
 items: [
 { text: 'Beds' },
 { text: 'Mattresses' },
 { text: 'Dining Tables' },
],
 },
 {
 text: 'Decor',
 items: [
 { text: 'Clocks' },
 { text: 'Wall Decals' },
 { text: 'Paintings' },
],
 },
],
},
];
let toolbarObj: Toolbar = new Toolbar({
 items: [
 { text: "Cut" },
 { text: "Copy" },
 { template: menuTemplate },
 { text: "Paste" }
],
});
toolbarObj.appendTo('#element');
let menuObj: Menu = new Menu({items: data,, '#menu'});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Accessibility in EJ2 JavaScript Toolbar control

The [JavaScript Toolbar](#) control has been designed, keeping in mind the [WAI-ARIA](#) specifications, and applying the WAI-ARIA roles, states, and properties along with keyboard support for people who use assistive devices. WAI-ARIA accessibility support is achieved through attributes like `aria-label`, and `aria-orientation`. It provides information about elements in a document for assistive technology. The control implements keyboard navigation support by following the [WAI-ARIA practices](#), and has been tested in major screen readers.

The accessibility compliance for the Toolbar control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

### ARIA attributes

Toolbar control is designed by considering [WAI-ARIA](#) standard. Toolbar is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices. The following list of attributes are added in the Toolbar.

#### | Property | Functionalities |

| --- | --- |

| role="toolbar" | Attribute is set to the ToolBar element describes the actual role of the element. |

| aria-orientation | Attribute is set to the ToolBar element to indicates the ToolBar orientation. Default value is **horizontal**. |

| aria-label | Attribute is set to ToolBar element describes the purpose of the set of toolbar. |

| aria-expanded | Attribute is set to the ToolBar Popup element to indicates the expanded state of the popup. |

| aria-haspopup | Attribute is set to the popup element to indicates the popup mode of the Toolbar. Default value is false. When popup mode is enabled, attribute value has to be changed to **true**. |

| aria-disabled | Attribute set to the ToolBar element to indicates the disabled state of the Toolbar. |

### Keyboard interaction

Keyboard navigation is enabled by default. Possible keys are

Key	Description
Left	Focuses the previous element.
Right	Focuses the next element.



| Enter | When focused on a ToolBar command, clicking the key triggers the click of Toolbar element. When popup drop-down icon is focused, the popup opens. |

| Esc(Escape) | Closes popup. |

| Down | Focuses the next popup element. |

| Up | Focuses the previous popup element. |

| Home | Moves focus to the first Toolbar. |

| End | Moves focus to the last Toolbar. |

| Tab | To Move focus through the interactive elements. |

| Shift + Tab | To Move focus through the interactive elements. |

### Ensuring accessibility

The Toolbar control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Toolbar control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Toolbar control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

### Style in EJ2 JavaScript Toolbar control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

#### Customizing Toolbar

Use the following CSS to customize the Toolbar.

```
`css
.e-toolbar {
border: 5px solid rgb(173, 255, 47);
}
```

#### Customizing the Toolbar items

Use the following CSS to customize the items of Toolbar.

```
`css
.e-toolbar .e-toolbar-item {
background: #add8e6;
border: 1px solid #5a70cc;
}
```

Use the following CSS to customize the button in the items of Toolbar.

```
`css
.e-toolbar .e-tbar-btn {
background: #add8e6;
border: 1px solid #5a70cc;
}
`
```

#### Customizing Toolbar's item icon

Use the following CSS to customize the item icon of Toolbar control.

```
`css
.e-toolbar .e-tbar-btn .e-icons {
background: #185655;
color: #d7f9d4;
}
`
```

#### Customizing the hover state of Toolbar control

Use the following CSS to customize the toolbar item when hovering.

```
`css
.e-toolbar .e-tbar-btn:hover {
background: #c0e3a1;
border: 1px solid green;
}
`
```

#### Customizing selected item of Toolbar control

Use the following CSS to customize the selected toolbar item.

```
`css
.e-toolbar .e-tbar-btn:focus {
background: #add8e6;
border: 1px solid #5a70cc;
}
`
```

## How To

Set command customization in EJ2 JavaScript Toolbar control

The [htmlAttributes](#) property of the Toolbar item is used to set the HTML attributes ('ID', 'class', 'style', 'role') for the commands.

When style attributes are added, if the same attributes were already present, they will be replaced. This is not so in the case of `class` attribute. Classes will be added to the element instead of replacing the existing ones.

Single or multiple CSS classes can be added to the Toolbar commands using the Toolbar item [cssClass](#) property.

### INDEX.TS

```
import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 width: 300,
 items: [
 { text: "Bold", type: 'Button', htmlAttributes: { 'class':
'custom_bold', 'id': 'itemId' } },
 { text: "Italic", htmlAttributes: { 'class': 'custom_italic' } },
 { text: "Underline", htmlAttributes: { 'class': 'custom_underline' }
 },
 { type: "Separator" },
 { text: "Uppercase", cssClass: "e-txt-casing" }
]
});
toolbar.appendTo('#element');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 <div id="result"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Set tool tip to the commands in EJ2 JavaScript Toolbar control

The [tooltipText](#) property of the Toolbar item is used to set the HTML Tooltip to the commands that can be viewed as hint texts on mouse hover.

To change the [tooltipText](#) to ej2-tooltip component:

- Import the **Tooltip** module from **ej2-popups**, and initialize the Tooltip with the Toolbar target. Refer to the following code example:

### INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
import { Toolbar } from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 width: 300,
 items: [
 { text: "Cut", tooltipText: 'Cut' },
 { text: "Copy", tooltipText: 'Copy' },
 { text: "Paste", tooltipText: 'Paste' },
 { text: "Undo", tooltipText: 'Undo' },
 { text: "Redo", tooltipText: 'Redo' }
]
});
toolbar.appendTo('#element');
let tooltip: Tooltip = new Tooltip({
 target: '#element [title]',
});
tooltip.appendTo('#Tooltip');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>

 <div id="container">
 <div id="Tooltip"><div id="element"></div></div>

 <div id="result"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Set item wise custom template in EJ2 JavaScript Toolbar control

The Toolbar supports adding template commands using the [template](#) property. Template property can be given as the **HTML element** that is either a **string** or a **query selector**.

#### As a string

The HTML element tag can be given as a string for the template property. Here, the checkbox is rendered as a HTML template.

```
`ts
```

```
template: "<div><input type='checkbox' id='check1' checked=''>Accept</input></div>"
```

```
,
```

#### As a selector

The template property also allows getting template content through query **selector**. Here, checkbox 'ID' attribute is specified in the template.

```
`ts
```

```
template: "#Template"
```

```
,
```

### INDEX.TS

```

import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 items: [
 { text: "Cut" },
 { type: "Separator" },
 { text: "Paste" },
 { type: "Separator" },
 { template: "<div><input type='checkbox' id='check1' checked=''>Accept</input></div>" },
 { text: "Undo" },
 { text: "Redo" },
 { template: "#Template" }
]
});

```

```
toolbar.appendTo('#element');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 <button id="Template" class="e-btn">Template</button>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Add font awesome in EJ2 JavaScript Toolbar control

You can customize the Toolbar component items by using third-party icons other than the icons available in the Syncfusion library. In the following example, font awesome icons are used as toolbar items.

- Refer to the third-party reference link. Here, the CDN link of font awesome is referred.

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome/4.7.0/css/font-
awesome.min.css" />
```

- Add the icons to the toolbar component using '[prefixIcon](#)' property

The following sample explains how to use font awesome in the toolbar component.

### INDEX.TS

```
import { Toolbar } from "@syncfusion/ej2-navigations";
let toolbar: Toolbar = new Toolbar({
 items: [
 { prefixIcon: "fa fa-twitter" },
 { prefixIcon: "fa fa-facebook" },
 { prefixIcon: "fa fa-whatsapp" },
 { template: '<button class="e-btn e-tbar-btn"><i class="e-icons fa fa-twitter"></i></button>' }
]
});
toolbar.appendTo("#element");
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <!-- <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet" /> -->

 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 <div id="result"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

We can also use templates for rendering icons based on the requirements.

#### Customization

The class “e-icons” is used for standardizing the appearance of the icons to fit into toolbar items. If you wish to override the appearance of the icons used, you could do this by using the following set of classes

Use the following CSS to set the color of icons.

```
`css
.e-toolbar .e-icons {
color: #e3165b !important;
}
`
```

Use the following CSS to set the font size of icons.

```
`css
.e-toolbar .e-btn .e-icons.e-btn-icon {
font-size: 14px !important;
}
`
```

#### Add toggle button in EJ2 JavaScript Toolbar control

JavaScript Toolbar supports to add a toggle Button by using the template property. Refer below steps

- By using Toolbar template property, pass required HTML String to render toggle button.

```
`ts
{ template: '<button class="e-btn" id="media_btn"></button>' }
`
```

- Now render the toggle Button into the targeted element in JavaScript Toolbar [created](#) event handler and bind click event for it. On clicking the toggle Button, change the required icon and content based on current active state.

#### INDEX.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { Toolbar } from '@syncfusion/ej2-navigations';
let undoBtn: any;
let zoomBtn: any;
let mediaBtn: any;
let filterBtn: any;
let visibleBtn: any;
let toolbar: Toolbar = new Toolbar({
 created: create,
 items: [
```



```

 { template: '<button class="e-btn" id="media_btn"></button>' },
 { type: "Separator" },
 { template: '<button class="e-btn" id="zoom_btn"></button>' },
 { type: "Separator" },
 { template: '<button class="e-btn" id="undo_btn"></button>' },
 { type: "Separator" },
 { template: '<button class="e-btn" id="filter_btn"></button>' },
 { type: "Separator" },
 { template: '<button class="e-btn" id="visible_btn"></button>' },
]
});
toolbar.appendTo('#element');
function create() {
 zoomBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-zoomin-
icon', isToggle: true });
 zoomBtn.appendTo('#zoom_btn');
 mediaBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-play-
icon', isToggle: true });
 mediaBtn.appendTo('#media_btn');
 undoBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-undo-
icon', isToggle: true });
 undoBtn.appendTo('#undo_btn');
 filterBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-filter-
icon', isToggle: true });
 filterBtn.appendTo('#filter_btn');
 visibleBtn = new Button({ cssClass: `e-flat`, iconCss: 'e-icons e-hide-
icon', isToggle: true, content: 'Hide' });
 visibleBtn.appendTo('#visible_btn');
 //Toggle button click event handlers
 zoomBtn.element.onclick = (): void => {
 if (zoomBtn.element.classList.contains('e-active')) {
 zoomBtn.iconCss = 'e-icons e-zoomout-icon';
 } else {
 zoomBtn.iconCss = 'e-icons e-zoomin-icon';
 }
 };
 mediaBtn.element.onclick = (): void => {
 if (mediaBtn.element.classList.contains('e-active')) {
 mediaBtn.iconCss = 'e-icons e-pause-icon';
 } else {
 mediaBtn.iconCss = 'e-icons e-play-icon';
 }
 };
 undoBtn.element.onclick = (): void => {
 if (undoBtn.element.classList.contains('e-active')) {
 undoBtn.iconCss = 'e-icons e-redo-icon';
 } else {
 undoBtn.iconCss = 'e-icons e-undo-icon';
 }
 };
 filterBtn.element.onclick = (): void => {
 if (filterBtn.element.classList.contains('e-active')) {
 filterBtn.iconCss = 'e-icons e-filternone-icon';
 } else {
 filterBtn.iconCss = 'e-icons e-filter-icon';
 }
 };
};

```

```

visibleBtn.element.onclick = (): void => {
 if (visibleBtn.element.classList.contains('e-active')) {
 document.getElementById('content').style.display = 'none';
 visibleBtn.content = 'Show';
 visibleBtn.iconCss = 'e-icons e-show-icon';
 } else {
 document.getElementById('content').style.display = 'block';
 visibleBtn.content = 'Hide';
 visibleBtn.iconCss = 'e-icons e-hide-icon';
 }
};
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element" class="toggle"></div>

 <div id="content">
 This content will be hidden, when you click on hide button and
toggle get an active state as show, otherwise it will be visible.
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Enable or disable toolbar item in EJ2 JavaScript Toolbar control

The [disabled](#) property of the Toolbar item is used to enable/disable the item by setting false/true value to the property. In the following code example initially paste action will be disabled. On clicking the cut or copy button, the paste button will be enabled.

**INDEX.TS**

```

import { Toolbar } from '@syncfusion/ej2-navigations';
function onClick() {
 toolbarObj.items[2].disabled = false;
}
let toolbarObj: Toolbar = new Toolbar({
 width: 600,
 items: [
 {
 prefixIcon: 'e-cut-icon tb-icons', tooltipText: 'Cut',
click: onClick},
 {
 prefixIcon: 'e-copy-icon tb-icons', tooltipText: 'Copy',
click: onClick},
 {
 prefixIcon: 'e-paste-icon tb-icons', tooltipText: 'Paste',
disabled: true },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-bold-icon tb-icons', tooltipText: 'Bold' },
 {
 prefixIcon: 'e-underline-icon tb-icons', tooltipText:
'Underline'},
 {
 prefixIcon: 'e-italic-icon tb-icons', tooltipText: 'Italic'
},
 {
 prefixIcon: 'e-color-icon tb-icons', tooltipText: 'Color-
Picker' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-alignleft-icon tb-icons', tooltipText:
'Align-Left' },
 {
 prefixIcon: 'e-alignjustify-icon tb-icons', tooltipText:
'Align-Justify'},
 {
 prefixIcon: 'e-alignright-icon tb-icons', tooltipText:
'Align-Right' },
 {
 prefixIcon: 'e-aligncenter-icon tb-icons', tooltipText:
'Align-Center' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-bullets-icon tb-icons', tooltipText:
'Bullets'},
 {
 prefixIcon: 'e-numbering-icon tb-icons', tooltipText:
'Numbering' },
 {
 type: 'Separator' },
 {

```

```

 prefixIcon: 'e-ascending-icon tb-icons', tooltipText: 'Sort
A - Z' },
 {
 prefixIcon: 'e-descending-icon tb-icons', tooltipText: 'Sort
Z - A' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-upload-icon tb-icons', tooltipText: 'Upload'
 },
 {
 prefixIcon: 'e-download-icon tb-icons', tooltipText:
'Download' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-indent-icon tb-icons', tooltipText: 'Text
Indent' },
 {
 prefixIcon: 'e-outdent-icon tb-icons', tooltipText: 'Text
Outdent' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-clear-icon tb-icons', tooltipText: 'Clear' },
 {
 prefixIcon: 'e-reload-icon tb-icons', tooltipText: 'Reload'
 },
 {
 prefixIcon: 'e-export-icon tb-icons', tooltipText: 'Export'
 }]
 });
 toolbarObj.appendTo('#element');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

 <div id="container">
 <div id="element"></div>

 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### How to customize toolbar scroll step in EJ2 JavaScript Toolbar control

Toolbar supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize **ScrollStep** property for scrolling distance. Refer to the following code example.

- By using Toolbar scrollStep property, pass a required value to customize toolbar scrollStep.

### INDEX.TS

```

import { Toolbar, BeforeCreateArgs } from '@syncfusion/ej2-navigations';
let toolbarObj: Toolbar = new Toolbar({
 scrollStep: 50,
 width: 600,
 items: [
 {
 prefixIcon: 'e-cut-icon tb-icons', tooltipText: 'Cut' },
 {
 prefixIcon: 'e-copy-icon tb-icons', tooltipText: 'Copy' },
 {
 prefixIcon: 'e-paste-icon tb-icons', tooltipText: 'Paste' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-bold-icon tb-icons', tooltipText: 'Bold' },
 {
 prefixIcon: 'e-underline-icon tb-icons', tooltipText:
'Underline' },
 {
 prefixIcon: 'e-italic-icon tb-icons', tooltipText: 'Italic'
 },
 {
 prefixIcon: 'e-color-icon tb-icons', tooltipText: 'Color-
Picker' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-alignleft-icon tb-icons', tooltipText:
'Align-Left' },
 {
 prefixIcon: 'e-alignjustify-icon tb-icons', tooltipText:
'Align-Justify'},
 {

```

```

 prefixIcon: 'e-alignright-icon tb-icons', tooltipText:
'Align-Right' },
 {
 prefixIcon: 'e-aligncenter-icon tb-icons', tooltipText:
'Align-Center' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-bullets-icon tb-icons', tooltipText:
'Bullets'},
 {
 prefixIcon: 'e-numbering-icon tb-icons', tooltipText:
'Numbering' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-ascending-icon tb-icons', tooltipText: 'Sort
A - Z' },
 {
 prefixIcon: 'e-descending-icon tb-icons', tooltipText: 'Sort
Z - A' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-upload-icon tb-icons', tooltipText: 'Upload'
 },
 {
 prefixIcon: 'e-download-icon tb-icons', tooltipText:
'Download' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-indent-icon tb-icons', tooltipText: 'Text
Indent' },
 {
 prefixIcon: 'e-outdent-icon tb-icons', tooltipText: 'Text
Outdent' },
 {
 type: 'Separator' },
 {
 prefixIcon: 'e-clear-icon tb-icons', tooltipText: 'Clear' },
 {
 prefixIcon: 'e-reload-icon tb-icons', tooltipText: 'Reload'
 },
 {
 prefixIcon: 'e-export-icon tb-icons', tooltipText: 'Export'
 }
]
});
toolbarObj.appendTo('#element');
function beforeCreate(e: BeforeCreateArgs) {
 e.scrollStep = 50;
};

```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 Toolbar</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Toolbar Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
<link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>

 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Add link to toolbar item in EJ2 JavaScript Toolbar control

Toolbar supports to add link by using the template configuration. The Toolbar can be rendered by item based collection and by HTML elements. Template property can be given as the **HTML element** that is either a **string** or a **query selector**.

The template property also allows getting template content through query **selector**. Here, anchor element 'ID' attribute is specified in the template.

```
`ts
```

```
template: "#AnchorTemplate"
```

```
,
```

### INDEX.TS

```

import {Toolbar} from '@syncfusion/ej2-navigations';
let toolbar: Toolbar = new Toolbar({
 items: [
 { text: "Cut" },
 { text: "Copy" },
 { type: "Separator" },
 { text: "Paste" },
]
});

```

```

 { type: "Separator" },
 { template: '#AnchorTemplate' }
]
});
toolbar.appendTo('#element');
```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Toolbar</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Toolbar Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/material.css"
rel="stylesheet">
 <link
href="https://ej2.syncfusion.com/angular/demos/src/toolbar/toolbar.component
.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="element"></div>
 <div id="AnchorTemplate">
 <a target="_blank"
href="https://ej2.syncfusion.com/documentation/toolbar/getting-
started/">Anchor Toolbar Link
 </div>

 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## Ej1 api migration in EJ2 JavaScript Toolbar control

This article describes the API migration process of Toolbar component from Essential JS 1 to Essential JS 2.

### Accessibility

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |



| Localization | **Not Applicable** | **Property:** *locale*   
 <br /> <br /> var toolbar = new ej.navigations.Toolbar({  
 <br /> &#160; locale: 'en-US' <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| RTL | **Property:** *enableRTL*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; enableRTL: true <br /> }); <br /> | **Property:** *enableRtl*   
 <br /> <br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; enableRtl: true <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

## DataSource

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| DataSource | **Property:** *dataSource*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; dataSource: items <br /> }); <br /> | **Not Applicable** |

| Query | **Property:** *query*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; query: query <br /> }); <br /> | **Not Applicable** |

| Fields | **Property:** *fields*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: {} <br /> }); <br /> | **Not Applicable** |

| Group | **Property:** *group*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { group: " " } <br /> }); <br /> | **Not Applicable** |

| HtmlAttributes | **Property:** *htmlAttributes*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { htmlAttributes: {} } <br /> }); <br /> | **Not Applicable** |

| Id | **Property:** *id*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { id: " " } <br /> }); <br /> | **Not Applicable** |

| ImageAttributes | **Property:** *imageAttributes*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { imageAttributes: {} } <br /> }); <br /> | **Not Applicable** |

| ImageUrl | **Property:** *imageUrl*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { imageUrl: " " } <br /> }); <br /> | **Not Applicable** |

| SpriteCssClass | **Property:** *spriteCssClass*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { spriteCssClass: " " } <br /> }); <br /> | **Not Applicable** |

| Text | **Property:** *text*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { text: " " } <br /> }); <br /> | **Not Applicable** |

| TooltipText | **Property:** *tooltipText*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { tooltipText: " " } <br /> }); <br /> | **Not Applicable** |

| Template | **Property:** *template*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; fields: { template: " " } <br /> }); <br /> | **Not Applicable** |

## Items

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** *items*   
 <br /> <br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [] <br /> }); <br /> | **Property:** *items*   
 <br /> <br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Align | **Not Applicable** | **Property:** *items[0].align* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> items: [{ align: 'center' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Custom Class | **Not Applicable** | **Property:** *items[0].cssClass* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> items: [{ cssClass: 'e-class' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Group Name | **Property:** *items[0].group* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ group: " " }] <br /> }); <br /> | **Not Applicable** |

| Html Attributes | **Property:** *items[0].htmlAttributes* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ htmlAttributes: {} }] <br /> }); <br /> | **Property:** *items[0].htmlAttributes* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ htmlAttributes: {} }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Id | **Property:** *items[0].id* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ id: " " }] <br /> }); <br /> | **Property:** *items[0].id* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ id: " " }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| ImageAttributes | **Property:** *items[0].imageAttributes* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ imageAttributes: {} }] <br /> }); <br /> | **Not Applicable** |

| ImageUrl | **Property:** *items[0].imageUrl* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ imageUrl: " " }] <br /> }); <br /> | **Not Applicable** |

| Overflow | **Not Applicable** | **Property:** *items[0].overflow* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ overflow: 'popup' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| PrefixIcon | **Not Applicable** | **Property:** *items[0].prefixIcon* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ prefixIcon: 'e-icon' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| ShowAlwaysInPopup | **Not Applicable** | **Property:** *items[0].showAlwaysInPopup* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ showAlwaysInPopup: true }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| ShowTextOn | **Not Applicable** | **Property:** *items[0].showTextOn* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ showTextOn: 'popup' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Sprite CssClass | **Property:** *items[0].spriteCssClass* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ spriteCssClass: 'e-class' }] <br /> }); <br /> | **Not Applicable** |

| SuffixIcon | **Not Applicable** | **Property:** *items[0].suffixIcon* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ suffixIcon: 'e-icon' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Template | **Property:** *items[0].template* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ template: " " }] <br /> }); <br /> | **Property:** *items[0].template* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ template: " " }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Text | **Property:** *items[0].text* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ text: 'Cut' }] <br /> }); <br /> | **Property:** *items[0].text* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ text: 'Cut' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| TooltipText | **Property:** *items[0].tooltipText* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; items: [{ tooltipText: 'Cut' }] <br /> }); <br /> | **Property:** *items[0].tooltipText* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ tooltipText: 'Cut' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Type | **Not Applicable** | **Property:** *items[0].type* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ type: 'Button' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Width | **Not Applicable** | **Property:** *items[0].width* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; items: [{ width: '50' }] <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Disable Items | **Property:** *disabledItemIndices* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; disabledItemIndices: [] <br /> }); <br /> | **Method:** *enableItems(items, false)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.enableItems([], false); <br /> |

| Add Items | **Not Applicable** | **Method:** *addItem(items, index)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.addItem([], 0); <br /> |

| RemoveItem | **Method:** *removeItem(element)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.removeItem(ele); <br /> | **Method:** *removeItems(args)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.removeItems(0); <br /> |

| RemoveItemById | **Method:** *removeItemById(id)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.removeItemById('left'); <br /> | **Not Applicable** |

| Enable Item | **Method:** *enableItem(element)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.enableItem(ele); <br /> | **Method:** *enableItems(items, true)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.enableItems(items, true); <br /> |

| EnableItemById | **Method:** *enableItemById(id)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.enableItemById('left'); <br /> | **Not Applicable** |

| Disable Item | **Method:** *disableItem(element)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.disableItem(ele); <br /> | **Method:** *enableItems(items, false)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.enableItems([], false); <br /> |

| DisableItemById | **Method:** *disableItemById(id)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.disableItemById('left'); <br /> | **Not Applicable** |

| Show Item | **Not Applicable** | **Method:** *hideItem(index, false)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.hideItem(0, false); <br /> |

| Hide Item | **Not Applicable** | **Method:** *hideItem(index, true)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.hideItem(0, true); <br /> |

| SelectItem | **Method:** *selectItem(element)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.selectItem(ele); <br /> | **Not Applicable** |

| SelectItemById | **Method:** *selectItemById(id)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.selectItemById('left'); <br /> | **Not Applicable** |

| Deselect Item | **Method:** *deselectItem(element)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.deselectItem(ele); <br /> | **Not Applicable** |

| DeselectItemById | **Method:** *deselectItemById(id)* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.deselectItemById('left'); <br /> | **Not Applicable** |

| Clicked | **Not Applicable** | **Event:** *clicked* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; clicked: function(e: Event): void { } <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| itemHover | **Event:** *itemHover* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; itemHover: function(args) { } <br /> }); <br /> | **Not Applicable** |

| itemLeave | **Event:** *itemLeave* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; itemLeave: function(args) { } <br /> }); <br /> | **Not Applicable** |

### Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Custom class | **Property:** *cssClass* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; cssClass: 'customClass' <br /> }); <br /> | **Not Applicable** |

| Enabled | **Property:** *enabled* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; enabled: false <br /> }); <br /> | **Not Applicable** |

| EnableSeparator | **Property:** *enableSeparator* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; enableSeparator: false <br /> }); <br /> | **Not Applicable** |

| Height | **Property:** *height* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; height: 400 <br /> }); <br /> | **Property:** *height* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; height: 700 <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| HtmlAttributes | **Property:** *htmlAttributes* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; htmlAttributes: { } <br /> }); <br /> | **Not Applicable** |

| Hide | **Property:** *hide* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; hide: true <br /> }); <br /> | **Not Applicable** |

| Orientation | **Property:** *orientation* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; orientation: ej.Orientation.Horizontal <br /> }); <br /> | **Not Applicable** |

| OverflowModes | **Property:** *responsiveType* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; responsiveType: 'popup' <br /> }); <br /> | **Property:** *overflowMode* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; overflowMode: 'Popup' <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Persistence | **Not Applicable** | **Property:** *enablePersistence* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; enablePersistence: true <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Responsive | **Property:** *isResponsive* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; isResponsive: true <br /> }); <br /> | **Not Applicable** |

| ShowRoundedCorner | **Property:** *showRoundedCorner* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; showRoundedCorner: false <br /> }); <br /> | **Not Applicable** |

| Width | **Property:** *width* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; width: 600 <br /> }); <br /> | **Property:** *width* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; width: 500 <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Enable | **Method:** *enable()* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.enable(); <br /> | **Method:** *disable(false)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.disable(false); <br /> |

| Disable | **Method:** *disable()* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.disable(); <br /> | **Method:** *disable(true)* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.disable(true); <br /> |

| Show | **Method:** *show()* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.show(); <br /> | **Not Applicable** |

| Hide | **Method:** *hide()* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.hide(); <br /> | **Not Applicable** |

| Refresh | **Not Applicable** | **Method:** *refresh()* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.refresh(); <br /> |

| Destroy | **Method:** *destroy()* <br /><br /> \$("#toolbar").ejToolbar(); <br /> var toolbar = \$("#toolbar").data("ejToolbar"); <br /> toolbar.destroy(); <br /> | **Method:** *destroy()* <br /><br /> var toolbar = new ej.navigations.Toolbar(); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> toolbar.destroy(); <br /> |

| Click | **Event:** *click* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; click: function(args) {} <br /> }); <br /> | **Not Applicable** |

| BeforeCreate | **Not Applicable** | **Event:** *beforeCreate* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; beforeCreate: function(e: Event): void {} <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Created | **Event:** *create* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; create: function(args) {} <br /> }); <br /> | **Event:** *created* <br /><br /> var toolbar = new ej.navigations.Toolbar({ <br /> &#160; created: function(e: Event): void {} <br /> }); <br /> toolbar.appendTo('#ej2Toolbar'); <br /> |

| Destroyed | **Event:** *destroy* <br /><br /> \$("#toolbar").ejToolbar({ <br /> &#160; destroy: function(args) {} <br /> }); <br /> | **Event:** *destroyed* <br /><br /> var toolbar = new

```
ej.navigations.Toolbar({
 destroyed: function(e: Event): void { }
 });

toolbar.appendTo('#ej2Toolbar');
 |
```

```
| FocusOut | Event: focusOut

 $(" #toolbar").ejToolbar({
 focusOut:
function(args) { }
 });
 | Not Applicable |
```

```
| overflowOpen | Event: overflowOpen

 $(" #toolbar").ejToolbar({

overflowOpen: function(args) { }
 });
 | Not Applicable |
```

```
| overflowClose | Event: overflowClose

 $(" #toolbar").ejToolbar({

overflowClose: function(args) { }
 });
 | Not Applicable |
```

## Tooltip

### Getting started in EJ2 JavaScript Tooltip control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

#### control Initialization

The Essential JS 2 JavaScript controls can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

#### *Using local script and style references in a HTML page*

**Step 1:** Create an app folder **myapp** for Essential JS 2 JavaScript controls.

**Step 2:** You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

#### **Syntax:**

Script: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE\_NAME}.min.js**

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

#### **Example:**

Script: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.17/Essential JS 2/ej2-popups/dist/global/ej2-popups.min.js**

Styles: **C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.17/Essential JS 2/ej2-popups/styles/material.css**

**Step 3:** Create a folder **myapp/resources** and copy/paste the global scripts and styles from the above installed location to **myapp/resources** location.

**Step 4:** Create a HTML page (index.html) in **myapp** location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Tooltip's global script -->
<script src="resources/ej2-popups.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

**Step 5:** Now, add the **Tooltip** element and initiate the **Essential JS 2 Tooltip** control in the **index.html** by using following code

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Tooltip's global script -->
<script src="resources/ej2-popups.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML span element -->
<div id='container'>
<div style="margin: 50px;">
Show Tooltip
</div>
</div>
`
```

```
<script>
// Initialize Essential JS 2 JavaScript Tooltip control
var tooltip = new ej.popups.Tooltip({
width: '150px',
height: '40px',
content: 'Tooltip with specific width and height'
});
tooltip.appendTo('#target');
</script>
</body>
</html>
`
```

**Step 6:** Now, run the `index.html` in web browser, it will render the **Essential JS 2 Tooltip** control.

*Using CDN link for script and style reference*

**Step 1:** Create an app folder `myapp` for the Essential JS 2 JavaScript controls.

**Step 2:** The Essential JS 2 control's global scripts and styles are already hosted in the below CDN link formats.

**Syntax:**

Script: `https://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `https://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

**Example:**

Script: <https://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js>

Styles: <https://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css>

We can also use [CRG](#) to generate combined control styles.

**Step 3:** Create a HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `Tooltip` element and initiate the **Essential JS 2 Tooltip** control in the `index.html` by using following code.

**INDEX.HTML**

```
<html><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```



```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head><body><script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

**Step 4:** Now, run the `index.html` in web browser, it will render the **Essential JS 2 Tooltip** control.

See Also

[Positioning Tooltip](#)

[Tooltip Open Mode](#)

[Customize the Tooltip](#)

## Setting dimension in EJ2 JavaScript Tooltip control

### Height and width

The Tooltip can either be assigned auto height and width values or specific pixel values. The `width` and `height` properties are used to

set the outer dimension of the Tooltip element. The default value for both the properties is `auto`. It also accepts string and number values in pixels.

The following sample explains how to set dimensions for the Tooltip.

### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let tooltip: Tooltip = new Tooltip({
 width: '180px',
 height: '40px',
 content: 'This tooltip has 180px width and 40px height'
});
tooltip.appendTo('#target');
let button: Button = new Button({
 content: 'Show Tooltip'
});
button.appendTo('#target');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div style="display: inline-block; position: relative; left:
50%;top: 100px;transform: translateX(-50%);">
 Show Tooltip
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Scroll mode

When **height** is specified with a certain pixel value and the Tooltip content overflows, the scrolling mode gets enabled.

### INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
let tooltipContent: HTMLElement = document.createElement("div");
tooltipContent.id = 'tooltipContent';
tooltipContent.innerHTML = "Environmentally friendly or environment-
friendly, (also referred to as eco-friendly, nature-friendly, and green) are
marketing and sustainability terms referring to goods and services, laws,
guidelines and policies that inflict reduced, minimal, or no harm upon
ecosystems or the environment.";
let tooltip: Tooltip = new Tooltip({
 width: '300px',
 height: '60px',
 content: tooltipContent,
 isSticky: true
});
tooltip.appendTo('#target');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <p>A green home is a type of house designed to be

 <u>environmentally friendly</u>
 and sustainable. And also focuses on the efficient use of
"energy, water, and building materials." As green homes
 have become more prevalent we have also seen the emergence of
green affordable housing.
 </p>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The scrolling mode can best be seen when the sticky mode of the Tooltip is enabled. To enable sticky mode, set the `isSticky` property to true.

### Content in EJ2 JavaScript Tooltip control

A text or a piece of information assigned to the Tooltip's `content` property will be displayed as the main text stream of the Tooltip.

It can be a string or a template content. If the `content` property is not provided with any specific value, then it takes the value

assigned to the `title` attribute of the target element on which the Tooltip was initialized. The content can also dynamically be

assigned to the Tooltip via AJAX.

### Template content

Any text or image can be added to the Tooltip, by default. To customize the Tooltip layout or to create your own visualized element on the

Tooltip, template can be used.

Refer to the following code example to add formatted HTML content to the Tooltip.

#### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
let tooltipContent: HTMLElement = document.createElement("div");
tooltipContent.innerHTML = "<p>Environmentally friendly or
environment-friendly, <i>(also referred to as eco-friendly,
nature-friendly, and green)</i> are marketing and sustainability terms
referring to goods and services, laws, guidelines and policies that inflict
reduced, minimal, or no harm upon ecosystems or the environment.</p>";
let tooltip: Tooltip = new Tooltip({
 content: tooltipContent
});
tooltip.appendTo('#target');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <p>A green home is a type of house designed to be

 <u>environmentally friendly</u>
 and sustainable. And also focuses on the efficient use of
 "energy, water, and building materials." As green homes
 have become more prevalent we have also seen the emergence of
 green affordable housing.
 </p>
 </div>
 <script>
 var ele = document.getElementById('container');
 if(ele) {
 ele.style.visibility = "visible";
 }
 </script>
 <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

### Dynamic content via AJAX

The Tooltip content can be dynamically loaded by making use of the AJAX call. The AJAX request is usually made within the `beforeRender`

event of the Tooltip, and then the Tooltip's `content` is assigned the value retrieved on it's success.

**Note:** The Tooltip **target** property includes a unique identifier used to associate Tooltips with specific elements on a webpage or application interface. When setting the Tooltip **target** value as a GUID (Globally Unique Identifier), it's important to note that the GUID must start with a combination of **letters** before the numeric portion of the GUID. For example, **target: '# + 'tooltip'+ '96ad88bd-294c-47c3-999b-a9daa3285a05'.**

### INDEX.TS

```
import { Tooltip, TooltipEventArgs } from '@syncfusion/ej2-popups';
import { Ajax } from '@syncfusion/ej2-base';
let tooltip: Tooltip = new Tooltip({
 content: 'Loading...',
 target: '.target',
 position: 'RightCenter',
 beforeRender: onBeforeRender
});
tooltip.appendTo('#targetContainer');
function onBeforeRender(args: TooltipEventArgs): void {
 let ajax: Ajax = new Ajax('./tooltipdata.json', 'GET', true);
 ajax.send().then(
 (result: any) => {
 result = JSON.parse(result);
 for (let i: number = 0; i < result.length; i++) {
 if (result[i].Id === args.target.getAttribute('data-
content')) {
 this.content = "<div class='contentWrap'><div>" +
result[i].Sports + "</div></div>";
 }
 }
 this.dataBind();
 },
 (reason: any) => {
 this.content = reason;
 this.dataBind();
 }
);
}
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <h4>National Sports</h4>
 <div id="targetContainer" class="e-prevent-select">
 <div id="countrylist">

 <li class="target" title="1">Australia
 <li class="target" title="2">Bhutan
 <li class="target" title="3">China
 <li class="target" title="4">Cuba
 <li class="target" title="5">India
 <li class="target"
title="6">Switzerland
 <li class="target" title="7">United
States

 </div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Position in EJ2 JavaScript Tooltip control

Tooltips can be attached to 12 static locations around the target. On initializing the Tooltip, you can set the position property with any one of the following values:

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter
- BottomRight
- LeftTop
- LeftCenter

- LeftBottom
- RightTop
- RightCenter
- RightBottom

By default, Tooltip is placed at the **TopCenter** of the target element.

#### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Button } from '@syncfusion/ej2-buttons';
let button: Button = new Button({content: 'Show Tooltip'});
button.appendTo('#tooltip');
let tooltip: Tooltip = new Tooltip({
 position: 'TopCenter',
 content: 'Tooltip Content'
});
tooltip.appendTo('#tooltip');
let dropDownListObject: DropDownList = new DropDownList({
 change: dropChange
});
dropDownListObject.appendTo('#positions');
function dropChange() {
 tooltip.close();
 tooltip.position = this.value;
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

<div id="container">
 <div id="tooltip"></div>
 <div class="ddl">
 <select id="positions" class="form-control" style="width:150px">
 <option value="TopLeft">Top Left</option>
 <option value="TopCenter" selected="">Top Center</option>
 <option value="TopRight">Top Right</option>
 <option value="BottomLeft">Bottom Left</option>
 <option value="BottomCenter">Bottom Center</option>
 <option value="BottomRight">Bottom Right</option>
 <option value="LeftTop">Left Top</option>
 <option value="LeftCenter">Left Center</option>
 <option value="LeftBottom">Left Bottom</option>
 <option value="RightTop">Right Top</option>
 <option value="RightCenter">Right Center</option>
 <option value="RightBottom">Right Bottom</option>
 </select>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Tip pointer positioning

The Tooltip pointer can be attached or detached from the Tooltip by using the `showTipPointer` property. Pointer positions can be adjusted using the `tipPointerPosition` property that can be assigned to one of the following values:

- Auto
- Start
- Middle
- End

The following code example illustrates how to set the pointer to the start position of the Tooltip.

#### INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let button: Button = new Button({content: 'Show Tooltip'});
button.appendTo('#target');
let tooltip: Tooltip = new Tooltip({
 tipPointerPosition: "Start"
});
tooltip.appendTo('#target');

```

#### INDEX.HTML



```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target" title="Tooltip content">
 Show tooltip
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, tip pointers are auto adjusted so that the arrow does not point outside the target element.

### Dynamic positioning

The Tooltip and its tip pointer can be positioned dynamically based on the target's location. This can be achieved by using the **refresh**

method, which auto adjusts the Tooltip over the target.

### INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
import { Draggable } from '@syncfusion/ej2-base';
let tooltip: Tooltip;
tooltip = new Tooltip({
 content: 'Drag me !!!',
 target: '#demoSmart',
 animation: { open: { effect: 'None' }, close: { effect: 'None' } }
}, '#targetContainer');
let ele: HTMLElement = document.getElementById('demoSmart');
let drag: Draggable = new Draggable(ele, {

```

```

clone: false,
dragArea: '#targetContainer',
drag: (args: any) => {
 tooltip.refresh(args.element);
},
dragStart: (args: any) => {
 tooltip.open(args.element);
},
dragStop: (args: any) => {
 tooltip.close();
}
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="targetContainer">
 <div id="demoSmart">
 </div>
 </div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Mouse trailing

Tooltips can be positioned relative to the mouse pointer. This behavior can be enabled or disabled by using the `mouseTrail` property. By default, it is set to `false`.

## INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 mouseTrail: true,
 content: 'Tooltip content'
});
tooltip.appendTo('#target');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target">
 Show tooltip
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

When mouse trailing option is enabled, the tip pointer position gets auto adjusted based on the target, and

other position values like start, end, and middle are not applied (to prevent the pointer from moving out of target).

### Setting offset values

Offset values are set to specify the distance between the target and tooltip element. `offsetX` and `offsetY` properties are used to specify the offset left and top values.

- `offsetX` specifies the distance between the target and Tooltip element in X axis.

- **offsetY** specifies the distance between the target and Tooltip element in Y axis.

The following code example illustrates how to set offset values.

#### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 offsetX: 30,
 offsetY: 30,
 mouseTrail: true,
 showTipPointer: false,
 content: 'Tooltip content'
});
tooltip.appendTo('#target');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target">
 Show tooltip
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

By default, collision is handled automatically and therefore when collision is detected the Tooltip fits horizontally and flips vertically.

## Open mode in EJ2 JavaScript Tooltip control

You can decide the mode on which the Tooltip is to be opened on a page, i.e., on hovering, focusing, or clicking on the target elements.

On mobile devices, Tooltips appear when you tap and hold the element, even if the `opensOn` option is assigned with `Hover`.

Tooltips are also displayed as long as you continue to tap and hold the element. On lift, it disappears in the next 1.5 seconds.

If there is another action before that time ends, then the Tooltip disappears.

The `opensOn` property can take either a single or a combination of multiple values, separated by space from the following options. The table below explains how the Tooltip opens on both desktop and mobile based on the value(s) assigned to the `opensOn` property. By default, it takes `auto` value.

Values	Desktop	Mobile
<code>Auto</code>	Tooltip appears when you hover over the target or when the target element receives the focus.	Tooltip opens on tap and hold of the target element.
<code>Hover</code>	Tooltip appears when you hover over the target.	Tooltip opens on tap and hold of the target element.
<code>Click</code>	Tooltip appears when you click a target element.	Tooltip appears when you single tap the target element.
<code>Focus</code>	Tooltip appears when you focus (say through tab key) on a target element.	Tooltip appears with a single tap on the target element.
<code>Custom</code>	Tooltip is not triggered by any default action. So, you have to bind your own events and use either <code>open</code> or <code>close</code> public methods. Same as Desktop.	

To open the Tooltip for multiple actions, say while hovering over or clicking on a target element, the `opensOn` property can be assigned

with multiple values, separated by space as `Hover Click`.

`auto` value cannot be used with any combination for multiple values.

The following code example shows how to set the open mode for Tooltips.

### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
let hoverTooltip: Tooltip = new Tooltip({
 opensOn: 'Hover',
 content: 'Tooltip from hover'
});
hoverTooltip.appendTo('#tooltiphover');
let button: Button = new Button();
button.appendTo('#tooltiphover');
let clickTooltip: Tooltip = new Tooltip({
 opensOn: 'Click',
```

```

 content: 'Tooltip from click'
 });
 clickTooltip.appendTo('#tooltipclick');
 button.appendTo('#tooltipclick');
 let focusTooltip: Tooltip = new Tooltip({
 opensOn: 'Focus',
 content: 'Tooltip from focus'
 });
 focusTooltip.appendTo('#tooltipfocus');
 let customTooltip: Tooltip = new Tooltip({
 opensOn: 'Custom',
 content: 'Tooltip from custom mode'
 });
 customTooltip.appendTo('#tooltipcustom');
 button.appendTo('#tooltipopen');
 document.getElementById('tooltipopen').addEventListener("click", function ()
 {
 if (this.getAttribute("data-tooltip-id")) {
 customTooltip.close();
 } else {
 customTooltip.open(this);
 }
 });

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <table style="margin: 150px auto 0 auto;transform: translateY(-
50%);">
 <tbody>
 <tr>
 <td style="padding:25px">

```

```

 <div id="tooltiphover" class="blocks">
 Hover Me !(Default)
 </div>
 </td>
 <td style="padding:25px">
 <div id="tooltipclick" class="blocks">
 Click Me !
 </div>
 </td>
</tr>
<tr>
 <td style="padding:25px">
 <div class="">

 <input id="tooltipfocus" type="text"
placeholder="Focus and blur">

 </div>
 </td>
 <td style="padding:25px">
 <div id="tooltipcustom">
 <div>
 <input id="tooltipopen" type="button"
value="Click to open tooltip manually">
 </div>
 </div>
 </td>
</tr>
</tbody>
</table>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Custom open mode

Other than the above specified options, the **custom** mode makes the Tooltip appear on screen for user-defined custom actions such as

right-click, double-click, and so on. Here, the tooltip is not triggered by any default action, and you have to bind your own events and use either **open** or **close** public methods to show or hide the Tooltips.

The following code example shows how to define custom open mode for the Tooltip.

### INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 opensOn: 'custom',
 content: 'Tooltip from custom mode'
});

```

```

});
tooltip.appendTo('#box');
document.getElementById('box').addEventListener("dblclick", function () {
 if (this.getAttribute("data-tooltip-id")) {
 tooltip.close();
 } else {
 tooltip.open(this);
 }
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="box">
 Double-click to open Tooltip.
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Sticky mode

With this mode set to **true**, Tooltips can be made to show up on the screen as long as the close icon is pressed. In this mode, close icon is attached to the Tooltip located at the top right corner. This mode can be enabled or disabled using the **isSticky** property.

## INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({

```



```
isSticky: true,
content: 'Click close icon to close me'
});
tooltip.appendTo('#target');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target">
 Show tooltip
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## Open/Close Tooltip with delay

The Tooltips can be opened or closed after some delay by using the `openDelay` and `closeDelay` properties.

## INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 openDelay: 1000,
 closeDelay: 1000,
 content: 'Tooltip with delay'
});
tooltip.appendTo('#target');
```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target">
 Show tooltip
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

**Animation in EJ2 JavaScript Tooltip control**

To animate the Tooltip, a set of specific animation effects are available, and it can be controlled using the **animation** property. The animation property also allows you to set delay, duration, and various other effects of your choice.

**AnimationModel** is derived from base to apply the chosen animation effect, duration, etc. on Tooltips.

By default, Tooltip entrance occurs over 150 ms using the **ease-out** timing function. It exits also at 150 ms, but uses **ease-in** timing function.

**INDEX.TS**

```

import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 content: 'Tooltip content',
 animation: {
 open: { effect: 'ZoomIn', duration: 1000, delay: 0 },
 close: { effect: 'ZoomOut', duration: 500, delay: 0 }
 }
});

```

```
tooltip.appendTo('#target');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target">
 Show tooltip
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

The default animation effect for the Tooltip is set to **FadeIn** for its open action, and **FadeOut** for its close action. The default **duration** is set to 150 ms and **delay** is set to 0.

### Animation effects

The animation effects that are applicable to Tooltips are:

- FadeIn
- FadeOut
- FadeZoomIn
- FadeZoomOut
- FlipXDownIn
- FlipXDownOut
- FlipXUpIn
- FlipXUpOut
- FlipYLeftIn

- FlipYLeftOut
- FlipYRightIn
- FlipYRightOut
- ZoomIn
- ZoomOut
- None

When the **effect** is specified as **none**, no effect will be applied to the Tooltip, and animation is considered to be set to **off**.

Some of the above animation effects suits the Tooltip better when its tip pointer is hidden.

This can be achieved by setting the **showTipPointer** to false.

#### Animating via open/close methods

Animations can also be applied on Tooltips dynamically through **open** and **close** methods. These methods accept the animation model as an

optional parameter. If you pass **TooltipAnimationSettings**, animation takes this model; otherwise, it takes the values from the **animation** property. It is also possible to pass different animations for Tooltips on each target.

Refer to the code snippet below to apply animations using public methods.

#### INDEX.TS

```
import { Tooltip, TooltipAnimationSettings } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 content: 'Tooltip content',
 opensOn: 'custom'
});
tooltip.appendTo('#target');
document.getElementById('target').addEventListener("click", function () {
 if (this.getAttribute("data-tooltip-id")) {
 let closeAnimation: TooltipAnimationSettings = { effect: 'FadeOut',
duration: 1000 }
 tooltip.close(closeAnimation);
 } else {
 let openAnimation: TooltipAnimationSettings = { effect: 'FadeIn',
duration: 1000 }
 tooltip.open(this, openAnimation);
 }
});
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target">
 Show tooltip
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Apply transition

The transition effect can be applied on Tooltips by using the `beforeRender` event as given in the following work-around code example.

#### INDEX.TS

```

import { Tooltip, TooltipEventArgs } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 target: '.circletool',
 closeDelay: 1000,
 animation: { open: { effect: 'ZoomIn', duration: 500 }, close: { effect:
'ZoomOut', duration: 500 } },
 beforeRender: onBeforeRender,
 afterClose: onAfterClose
});
function onBeforeRender(args: TooltipEventArgs): void {
 if (args.element) {
 // here prevent animation while apply transition
 this.animation = { open: { effect: 'None' } };
 args.element.style.display = 'block';
 args.element.style.transitionProperty = 'left,top';
 args.element.style.transitionDuration = '1000ms';
 }
}
function onAfterClose(args: TooltipEventArgs): void {
 // restore the animation effects
 this.animation = { open: { effect: 'ZoomIn', duration: 500 }, close: {
effect: 'ZoomOut', duration: 500 } };
}
tooltip.appendTo('#box');

```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <h3> Transition effect </h3>
 <div id="box" class="e-prevent-select">
 <div class="circletool" style="top:18%;left:5%" title="This is
Turtle !!!"></div>
 <div class="circletool" style="top:30%;left:30%" title="This is
Snake !!!"></div>
 <div class="circletool" style="top:80%;left:80%" title="This is
Croc !!!"></div>
 <div class="circletool" style="top:65%;left:50%" title="This is
String Ray !!!"></div>
 <div class="circletool" style="top:75%;left:15%" title="This is
Blob Fish !!!"></div>
 <div class="circletool" style="top:30%;left:70%" title="This is
Mammoth !!!"></div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

**Customization in EJ2 JavaScript Tooltip control**

The Tooltip can be customized by using the `cssClass` property, which accepts custom CSS class names that define specific user-defined

styles and themes to be applied on the Tooltip element.

### Tip pointer customization

Styling the tip pointer's size, background, and border colors can be done using the `cssClass` property, as given below.

#### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
let tooltip: Tooltip = new Tooltip({
 cssClass: 'customtip'
});
tooltip.appendTo('#target');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target" title="Tooltip arrow customized">
 Show tooltip
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

### Tooltip customization

The complete look and feel of the Tooltip can be customized by changing its background color, opacity, content font, etc. The following code example shows the way to achieve it.

#### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
```

```
let tooltip: Tooltip = new Tooltip({
 cssClass: 'customtooltip'
});
tooltip.appendTo('#target');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="target" title="Tooltip customized">
 Show tooltip
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## Style in EJ2 JavaScript Tooltip control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

### Customizing the tooltip

Use the following CSS to customize the tooltip.

```
.e-tooltip-wrap {
border-radius: 4px;
opacity: 1;
```



```
}
,
```

### Customizing the tooltip popup

Use the following CSS to customize the tooltip popup properties.

```
,
```

```
.e-tooltip-wrap.e-popup {
background-color: #fff;
border: 2px solid #000;
}
,
```

### Customizing the tooltip content

Use the following CSS to customize the tooltip content.

```
,
```

```
.e-tooltip-wrap .e-tip-content {
color: red;
font-size: 12px;
line-height: 20px;
}
,
```

### Customizing the tooltip arrow tip

Use the following CSS to customize the tooltip arrow tip.

```
,
```

*/ To customize the arrow tip at bottom /*

```
.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
height: 12px;
left: 50%;
top: 100%;
width: 24px;
}

/ To customize the arrow tip at top /
```

*/ To customize the arrow tip at top /*

```
.e-tooltip-wrap .e-arrow-tip.e-tip-top {
height: 12px;
left: 50%;
top: -9px;
}
```

```
width: 24px;
}
/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip.e-tip-left {
height: 24px;
left: -9px;
top: 48%;
width: 12px;
}
/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip.e-tip-right {
height: 24px;
left: 100%;
top: 50%;
width: 12px;
}
,
```

#### Customizing the tooltip inner tip

Use the following CSS to customize the tooltip inner tip.

```
,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
color: #fff;
font-size: 25.9px;
}
,
```

#### Customizing the tooltip outer tip

Use the following CSS to customize the tooltip outer tip.

```
,
/ To customize the arrow tip at bottom /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-left: 12px solid transparent;
border-right: 14px solid transparent;
border-top: 12px solid #000;
}
,
```

```

/ To customize the arrow tip at top /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
border-bottom: 12px solid #000;
border-left: 12px solid transparent;
border-right: 12px solid transparent;
}

/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
border-bottom: 12px solid transparent;
border-right: 12px solid #000;
border-top: 12px solid transparent;
}

/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
border-bottom: 12px solid transparent;
border-left: 12px solid #000;
border-top: 12px solid transparent;
}
,

```

### Accessibility in EJ2 JavaScript Tooltip component

The Tooltip component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Tooltip component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

#### WAI-ARIA attributes

The Tooltip component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Tooltip component.

| Attributes | Description |

| --- | --- |

| role="tooltip" | The element that serves as the container for the tooltip has the ARIA role of **tooltip**. |

| aria-describedby | This attribute is added to the target element on which the Tooltip gets opened, when focusing or hovering over it. It usually holds the randomly generated Id value of the Tooltip element. <br /> <br />In case, the target element already holds an **aria-describedby** attribute with Id value of some other component, then the Tooltip Id value can be additionally appended to the existing **aria-describedby** attribute separated by a space as shown in the example below.<br /><br /> **For example:** <br /> aria-describedby = "my-text my-tooltip" <br /> **my-text** is the Id of some other component.<br /> **my-tooltip** is the id of Tooltip component. <br /><br /> When the Tooltip is closed, the **aria-describedby** attribute is removed from the target. |

| aria-hidden | This attribute is assigned to the Tooltip element whose default value is true. <br /><br /> When **true**, it denotes that the Tooltip element is in a hidden or a closed state. When the Tooltip appears on the screen, it's value changes to **false**. |

## Keyboard interaction

The Tooltip component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Tooltip component.

Keys	Description
Esc	Closes or dismisses the Tooltip.
Tab	A form control receiving focus (say through tab key), opens the Tooltip, and on focus out closes it.

1. When the Tooltip is being displayed on the target element, focus continues to stay on it.
2. If the Tooltip opens on mouse entering into the target element space, then it should be dismissed only when the mouse leaves that target.
3. If the Tooltip opens on the target element that receives focus, then it should be closed only when the focus moves out of that target element.

Likewise, if the Tooltip opens on a click, then it should be closed only on another click action.

## Ensuring accessibility

The Tooltip component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Tooltip component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Tooltip component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

## Ej1 api migration in EJ2 JavaScript Tooltip control

This article describes the API migration process of Tooltip component from Essential JS 1 to Essential JS 2

Behavior | API in Essential JS 1 | API in Essential JS 2

--- | --- | ---

Position | **Property:** *position* `<ej-tooltip position: {target: {horizontal: "center", vertical: "top"}, stem: {horizontal: "center", vertical: "bottom"}}>` | **Property:** *position* `let tooltip: Tooltip = new ej.popups.Tooltip({position: 'TopCenter'}); tooltip.appendTo('#test');`

Animation | **Property:** *animation* `<ej-tooltip animation: {effect: "slide", speed: 1000}>` | **Property:** *animation* `let tooltip: Tooltip = new ej.popups.Tooltip({position: 'TopCenter', animation: {open: {effect: 'FadeIn', duration: 150, delay: 0}, close: {effect: 'FadeOut', duration: 150, delay: 0}}}); tooltip.appendTo('#test');`

Close Time Out | **Property:** *autoCloseTimeout* `<ej-tooltip autoCloseTimeout: 5000>` | **Property:** *closeDelay, openDelay* `let tooltip: Tooltip = new`

```

ej.popups.Tooltip({
 position : 'TopCenter',
 closeDelay : 500
 });

tooltip.appendTo('#test'); |

| Sticky Mode | Property : closeMode
 $(("#test").ejTooltip
({
closeMode : "sticky"

}); | Property: isSticky
 let tooltip: Tooltip = new ej.popups.Tooltip({
 position :
'TopCenter',
 isSticky : true
 });
 tooltip.appendTo('#test'); |

| Offset from target | Property : tip.adjust.xValue/ tip.adjust.yValue
 $(("#test").ejTooltip
 { {

 tip :
 size : { width : 25, height : 12},
 adjust : {xValue : 5, yValue: 6}
}
 }; |
Property: offsetX/ offsetY
 let tooltip: Tooltip = new ej.popups.Tooltip({
position :
'TopCenter',
 offsetX : 10,
offsetY : 10
});
tooltip.appendTo('#test'); |

| Mouse trail on target | Not Applicable | Property: mouseTrail
 let tooltip: Tooltip = new
ej.popups.Tooltip({
mouseTrail: true,
});
tooltip.appendTo('#test'); |

| Open mode of tooltip | Not Applicable | Property: opensOn
 let tooltip: Tooltip = new
ej.popups.Tooltip({
opensOn: 'Click',
});
tooltip.appendTo('#test'); |

| Enable/disable the tip of tooltip | Not Applicable | Property: showTipPointer
 let tooltip:
Tooltip = new ej.popups.Tooltip({
 showTipPointer: false,
});
tooltip.appendTo('#test'); |

| Hide | Method: hide()
 var tooltip = $(("#test").ejTooltip({
 content: "JavaScript is the
programming language of HTML and the Web." }).data("ejTooltip");
 tooltip.hide(); | Method:
close()
 let tooltip: Tooltip = new ej.popups.Tooltip({
 cssClass: 'e-tooltip-css',

opensOn: 'Custom',
content: 'Tooltip from custom mode'
});
tooltip.appendTo('#test');

tooltip.close();|

| Show | Method: show()
 var tooltip = $(("#test").ejTooltip({
 content: "JavaScript is the
programming language of HTML and the Web." }).data("ejTooltip");
 tooltip.show(); | Method:
open()
 let tooltip: Tooltip = new ej.popups.Tooltip({
 cssClass: 'e-tooltip-css',
 content:
'Tooltip from custom mode'
});
tooltip.appendTo('#test');
tooltip.open();|

| Enable | Method: enable()
 var tooltip = $(("#test").ejTooltip({
 content: "JavaScript is the
programming language of HTML and the Web." }).data("ejTooltip");
 tooltip.enable(); | Method:
destroy()
 let tooltip: Tooltip = new ej.popups.Tooltip({
cssClass: 'e-tooltip-css',
content: 'Tooltip from custom mode'
});
tooltip.appendTo('#test');
tooltip.destory(); |

| Close | Event: close
 $(("#test").ejTooltip({
 content: "JavaScript is the programming
language of HTML and the Web.", close: function (args) { }
 }); | Event: afterClose
 let
tooltip: Tooltip = new ej.popups.Tooltip({
 afterClose: function(e: Event): void { }
 });

tooltip.appendTo('#test'); |

| Open | Event: open
 $(("#test").ejTooltip({
 content: "JavaScript is the programming
language of HTML and the Web.", open: function (args) { }
 }); | Event: afterOpen
 let tooltip:
Tooltip = new ej.popups.Tooltip({
 afterOpen: function(e: Event : void { }
 });

tooltip.appendTo('#test');

| Before Collision | Not Applicable | Event: beforeCollision
 let tooltip: Tooltip = new
ej.popups.Tooltip({
 beforeCollision: function(e: Event): void { }
});
tooltip.appendTo('#test'); |

| Tracking | Event: tracking
 $(("#test").ejTooltip({
 content: "JavaScript is the programming
language of HTML and the Web.", associate : "mouse",
 tracking: function (args) { }
 }); |
Method: open(),close(),refresh()
 let tooltip: Tooltip = new ej.popups.Tooltip({
content: 'Drag

```

```
me anywhere, to start walking with me !!!',
offsetX: -15,
target: '#demoSmart',
animation: { open: { effect: 'None' }, close: { effect: 'None' } }
});
tooltip.appendTo('#test');
let ele: HTMLElement = document.getElementById('demoSmart');
let
drag: Draggable = new Draggable(ele, {
clone: false,
dragArea: '#targetContainer',
drag: (args: any) => {
if (args.element.getAttribute('data-tooltip-id') === null) {
tooltip.open(args.element);
} else {
tooltip.refresh(args.element);
}
},
dragStart: (args: any) => {
if (args.element.getAttribute('data-tooltip-id') !== null) { return;
}
tooltip.open(args.element);
},
dragStop: (args: any) => {
tooltip.close();
}
});
 |
```

## How To

Show tooltip on disabled elements and disable tooltip in EJ2 JavaScript Tooltip control

By default, tooltips will not be displayed on disabled elements. However, you can enable this behavior by using the following steps:

1. Add a disabled element like the **button** element into a div whose display style is set to **inline-block**.
2. Set the pointer event as **none** for the disabled element (button) through CSS.
3. Initialize the tooltip for outer div element that holds the disabled button element.

## INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button, Switch } from '@syncfusion/ej2-buttons';
let tooltipObj_1: Tooltip = new Tooltip({
 content: 'Tooltip from disabled element'
});
tooltipObj_1.appendTo('#box');
//Initialize Button component
let buttonObj_1: Button = new Button({
 disabled: true
});
//Render initialized Button component
buttonObj_1.appendTo('#disabledbutton');
//Initialize Button component
let button: Button = new Button();
//Render initialized Button component
button.appendTo('#tooltip');
//Initialize Tooltip component
let tooltipObj_2: Tooltip = new Tooltip({
 //Set tooltip content
 content: 'Lets go green & Save Earth !!!'
});
//Render initialized Tooltip component
tooltipObj_2.appendTo('#tooltip');
let switchObj: Switch = new Switch({
 value: 'USB tethering',
 checked: true,
 change: (args) => {
 if ((document.getElementById('checked') as HTMLInputElement).checked) {
 let tooltipObj_2: Tooltip = new Tooltip({
 //Set tooltip content
 content: "Lets go green & Save Earth !!!"
 });
 }
 }
});
```

```

 });
 //Render initialized Tooltip component
 tooltipObj_2.appendTo('#tooltip');
 } else {
 tooltipObj_2.destroy();
 }
}
});
switchObj.appendTo('#checked');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="box" style="display: inline-block;">
 <input type="button" id="disabledbutton" value="Disabled
button">
 </div>
 <div style="position: relative;top: 75px;">
 <!-- Tooltip element -->
 <button id="tooltip">Show Tooltip</button>
 <div class="switchContainer">
 <label for="checked" style="padding: 0 25px 0 0">Enable
Tooltip</label>
 <input id="checked" type="checkbox">
 </div>
 </div>
 </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>

```



```
</body></html>
```

### Dynamic tooltip content with html in EJ2 JavaScript Tooltip control

The Tooltip component loads HTML tags using the [content](#) template.

The HTML tags such as `<div>`, `<span>`, **bold**, *italic*, underline, etc., can be used. Style attributes can also be applied with HTML tags.

Here, Bold, Italic, Underline, and Anchor tags are used.

When using HTML elements as content to **Tooltip** make the content element `display: none`, then from the [beforeRender](#) event we can make the element visible again using below code.

```
`ts
```

```
document.getElementById('content').style.display = 'block';
```

### INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
//Define an array of JSON data
let title: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 position: 'BottomCenter',
 opensOn: 'Hover',
 beforeRender: onBeforeRender,
 content: document.getElementById('tooltip')
});
title.appendTo('#Title');
let btn: Button = new Button();
btn.appendTo('#Title');
function onBeforeRender() {
 if (document.getElementById('tooltip')) {
 document.getElementById('tooltip').style.display = 'block';
 }
}
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="tooltip">
 <h2>HTML Tags</h2>
 Through templates, tooltip
content can be loaded with <u><i> inline HTML, images, iframe,
videos, maps </i></u>. A title can be added to the content</div>
 <div id="tooltipContent">
 <div class="content">
 <button class="text" id="Title">HTML(With Title)</button>
 </div>
 </div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Custom tooltip with dynamic html in EJ2 JavaScript Tooltip control

Tooltip loads HTML pages via HTML tags such as iframe, video, and map using the [content](#) property, which supports both string and HTML tags.

To load an `iframe` element in tooltip, set the required iframe in the `content` of tooltip while initializing the tooltip component. Refer to the following code.

```
`ts
```

```
content: '<iframe src="https://www.syncfusion.com/products/essential-js2"></iframe>
```

```
,
```

Use the following steps to render `ej2-map` in tooltip:

1. Initialize the map component and create an element. After initialization, append the map object to the element.
2. Set the rendered map element to the content of tooltip component. Refer to the following sample.

### INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
//Render iframe button
let iframeContent: Button = new Button({ cssClass: 'e-outline', isPrimary:
true });

```

```

iframeContent.appendTo('#iframeContent');
//Render iframe tooltip
let TooltipContent: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: '<iframe
src="https://ej2.syncfusion.com/showcase/typescript/expensetracker/#/dashboa
rd" id="tooltipFrame"></iframe>',
 position: 'BottomCenter',
 opensOn: 'Click',
 width: '350',
 height: '250'
});
TooltipContent.appendTo('#iframeContent');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for ListView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for ListView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
maps/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="tooltipContent">
 <div class="content">
 <!-- iframe element -->
 <button class="text" id="iframeContent">Embedded
Iframe</button>
 </div>
 </div>
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Tooltip open or display modes in EJ2 JavaScript Tooltip control

The open mode property of tooltip can be defined on a target that is hovering, focusing, or clicking.

Tooltip component have the following types of open mode:

- Auto
- Hover
- Click
- Focus
- Custom

#### Auto

Tooltip appears when you hover over the target or when the target element receives the focus.

#### Hover

Tooltip appears when you hover over the target.

#### Click

Tooltip appears when you click a target element.

#### Focus

Tooltip appears when you focus (say through tab key) on a target element.

#### Custom

Tooltip is not triggered by any default action. So, bind your own events and use either open or close public methods.

### **INDEX.TS**

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
//Render button components
let tooltiphover: Button = new Button({ cssClass: 'e-outline', isPrimary:
true });
tooltiphover.appendTo('#tooltiphover');
let tooltipclick: Button = new Button({ cssClass: 'e-outline', isPrimary:
true });
tooltipclick.appendTo('#tooltipclick');
let tooltipcustom: Button = new Button({ cssClass: 'e-outline', isPrimary:
true });
tooltipcustom.appendTo('#tooltipcustom');
let Mode: Button = new Button({ cssClass: 'e-outline', isPrimary: true });
Mode.appendTo('#Mode');
let open: Button = new Button({ cssClass: 'e-outline', isPrimary: true });
open.appendTo('#openMode');
//Render tooltip component
let hoverTooltip: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 opensOn: 'Hover',
 content: 'Tooltip from hover'
});
hoverTooltip.appendTo('#tooltiphover');
```

```

let clickTooltip: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 opensOn: 'Click',
 content: 'Tooltip from click'
});
clickTooltip.appendTo('#tooltipclick');
let focusTooltip: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 opensOn: 'Focus',
 content: 'Tooltip from focus'
});
focusTooltip.appendTo('#focus');
let customTooltip: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 opensOn: 'Custom',
 content: 'Tooltip from custom mode'
});
customTooltip.appendTo('#tooltipcustom');
let openMode: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 openDelay: 1000,
 closeDelay: 1000,
 position: 'BottomCenter',
 content: 'Opens and closes Tooltip with delay of <i>1000
milliseconds</i>'
});
openMode.appendTo('#openMode');
let mode: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'Click close icon to close me',
 position: 'BottomCenter',
 isSticky: true
});
mode.appendTo('#Mode');
document.getElementById('tooltipcustom').addEventListener("dblclick",
function () {
 if (this.getAttribute("data-tooltip-id")) {
 customTooltip.close();
 } else {
 customTooltip.open(this);
 }
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="showTooltip">
 <div id="first">
 <button class="blocks" id="tooltiphover">Hover
me! (Default)</button>
 <button class="blocks" id="tooltipclick">Click me!</button>
 </div>

 <div id="second">
 <button class="blocks" id="Mode">Sticky Mode</button>
 <button class="blocks" id="openMode">Tooltip with
delay</button>
 </div>

 <div id="third">
 <button class="blocks" id="tooltipcustom">Double click on
me!</button>
 <div id="textbox" class="e-float-input blocks">
 <input id="focus" type="text" placeholder="Focus and blur">
 </div>
 </div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Fancy tooltip customization in EJ2 JavaScript Tooltip control

The arrow of the tooltip can be customized as needed by customizing CSS in the sample-side. The EJ2 tooltip component is achieved through CSS3 format and positioned the tip arrow according to the tooltip positions like **TopCenter**, **BottomLeft**, **RightTop**, and more.

Here, the tip arrow is customized as Curved tooltip and Bubble tooltip.

#### Curved tip

The content for the tip pointer arrow has been added. To customize the curved tip arrow, override the following CSS class of tip arrow.

```
`ts
```

```
let tooltip: Tooltip = new Tooltip({
 cssClass: 'curvetips e-tooltip-css',
 content: 'Tooltip arrow customized',
});
tooltip.appendTo('#target');
```

```
`ts
.e-arrow-tip-outer.e-tip-bottom,
.e-arrow-tip-outer.e-tip-top {
 border-left: none !important;
 border-right: none !important;
 border-top: none !important;
}
.e-arrow-tip.e-tip-top {
 transform: rotate(170deg);
}
```

#### Bubble tip

The two **divs** (inner div and outer div) have been added to achieve the bubble tip arrow. To customize the bubble tip arrow, override the following CSS class of tip arrow.

```
`ts
let bubble: Tooltip = new Tooltip({
 cssClass: 'bubbletip e-tooltip-css',
 position: 'TopRight',
 content: 'Tooltip arrow customized as balloon tip'
});
bubble.appendTo('#bubbletip');
```

```
`ts
.e-arrow-tip-outer.e-tip-bottom, .e-arrow-tip-outer.e-tip-top
{
 border-radius: 50px;
 height: 10px;
 width: 10px;
```

```

}
.e-arrow-tip-inner.e-tip-bottom, .e-arrow-tip-inner.e-tip-top
{
border-radius: 50px;
height: 10px;
width: 10px;
}
`

```

These tip arrow customizations have been achieved through CSS changes in the sample level. The tooltip position can be changed by using the radio button click event.

The arrow tip pointer can also be disabled by using the [showTipPointer](#) property in a tooltip.

### INDEX.TS

```

import { Tooltip } from '@syncfusion/ej2-popups';
import { RadioButton, ChangeArgs, Button } from '@syncfusion/ej2-buttons';
let tooltip: Tooltip = new Tooltip({
 cssClass: 'curvetips e-tooltip-css',
 content: 'Tooltip arrow customized',
});
tooltip.appendTo('#target');
let curvebutton: Button = new Button();
curvebutton.appendTo('#target');
let tipbutton: Button = new Button();
tipbutton.appendTo('#tooltip');
let bubblebutton: Button = new Button();
bubblebutton.appendTo('#bubbletip');
let radiobutton: RadioButton = new RadioButton({ label: 'TopCenter', name:
'default', value: 'TopCenter', checked: true, change: onChange});
// Render initialized radio button
radiobutton.appendTo('#element1');
let radiobutton = new RadioButton({ label: 'BottomLeft', name: 'default',
value: 'BottomLeft', change: onChange});
radiobutton.appendTo('#element2');
let tippointer: Tooltip = new Tooltip({
 cssClass: 'pointertip e-tooltip-css',
 mouseTrail: true,
 content: 'Disabled tooltip pointer',
 showTipPointer: false
});
tippointer.appendTo('#tooltip');
let bubble: Tooltip = new Tooltip({
 cssClass: 'bubbletip e-tooltip-css',
 position: 'TopRight',
 content: 'Tooltip arrow customized as balloon tip'
});
bubble.appendTo('#bubbletip');
let radiobutton: RadioButton = new RadioButton({ label: 'BottomLeft', name:
'position', value: 'BottomLeft', change: onChange});
// Render initialized radio button
radiobutton.appendTo('#radiol');

```



```

let radiobutton = new RadioButton({ label: 'TopRight', name: 'position',
value: 'TopRight', checked: true, change: onChanged});
radiobutton.appendTo('#radio2');
function onChange(args: ChangeArgs): void {
 tooltip.position = args.value as any;
 tooltip.dataBind();
}
function onChanged(args: ChangeArgs): void {
 bubble.position = args.value as any;
 if(bubble.position == 'BottomLeft'){
 bubble.offsetY = -30;
 } else {
 bubble.offsetY = 0;
 }
 bubble.dataBind();
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="customization" class="customTipContainer">
 <button id="target">
 Customized Tip Arrow
 </button>
 <div id="positions">

 <input type="radio" id="element1">

 <input type="radio" id="element2">

 </div>
 </div>
 </div>

```

```

</div>
<div id="balloon" class="customTipContainer">
 <button id="bubbletip">
 Bubble Tip Arrow
 </button>
 <div id="btn">

 <input type="radio" id="radio1">

 <input type="radio" id="radio2">

 </div>
</div>
<div id="disabledContainer" class="customTipContainer">
 <button id="tooltip">
 Disabled Tip Arrow
 </button>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Display tooltip on svg and canvas elements in EJ2 JavaScript Tooltip control

Tooltip can be displayed on both SVG and Canvas elements. You can directly attach the `<svg>` or `<canvas>` elements to show tooltips on data visualization elements.

#### SVG

Create the SVG square element and refer to the following code snippet to render the tooltip on SVG square.

```
`ts
```

```

let square: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'SVG Square',
 target: '#square'
});
square.appendTo('#box');
`

```

*Canvas*

Create the canvas circle element and refer to the following code snippet to render the tooltip on Canvas circle.

```
`ts
```

```
let circle: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'Canvas Circle',
 target: '#circle'
});
circle.appendTo('#box');
```

**INDEX.TS**

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
//Render tooltip component
let square: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'SVG Square',
 target: '#square'
});
square.appendTo('#box');
let ellipse: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'SVG Ellipse',
 target: '#ellipse'
});
ellipse.appendTo('#box');
let polyline: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'SVG Polyline',
 target: '#polyline'
});
polyline.appendTo('#box');
let circle: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'Canvas Circle',
 target: '#circle'
});
circle.appendTo('#box');
let triangle: Tooltip = new Tooltip({
 cssClass: 'e-tooltip-css',
 content: 'Canvas Triangle',
 target: '#triangle'
});
triangle.appendTo('#box');
let canvas: HTMLCanvasElement = document.getElementById('triangle') as
HTMLCanvasElement;
let context;
if (canvas.getContext) {
```

```

 context = canvas.getContext('2d');
 context.beginPath();
 context.moveTo(0, 50);
 context.lineTo(100, 50);
 context.lineTo(50, 0);
 context.fillStyle = "#FDA600";
 context.fill();
}
canvas = document.getElementById('circle') as HTMLCanvasElement;
context = canvas.getContext('2d');
let centerX: number = canvas.width / 2;
let centerY: number = canvas.height / 2;
let radius: number = 30;
context.beginPath();
context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
context.fillStyle = '#0450C2';
context.fill();

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="box" class="e-prevent-select">
 <div class="circletool" id="rectShape" style="left:1%;top:10%">
 <svg>
 <rect id="square" class="shape" x="50" y="20" width="50"
height="50" style="fill:#FDA600;stroke:none;stroke-width:5;stroke-
opacity:0.9"></rect>
 </svg>
 </div>
 <div class="circletool" id="ellipseShape" style="top:65%;">
 <svg>
 <ellipse id="ellipse" class="shape" cx="100" cy="50" rx="20"
ry="40" style="fill:#0450C2;stroke:none;stroke-width:2"></ellipse>
 </svg>
 </div>
 </div>
 </div>

```

```

 </div>
 <div class="circletool" id="polyShape" style="top:25%;left:40%">
 <svg>
 <polyline id="polyline" class="shape" points="0,40 40,40
40,80 80,80 80,120 120,120 120,160"
style="fill:#ffffff;stroke:#0450C2;stroke-width:4"></polyline>
 </svg>
 </div>
 <div class="circletool" id="circleShape"
style="top:16%;left:72%">
 <canvas id="circle" class="shape" width="60"
height="60"></canvas>
 </div>
 <div class="circletool" id="triShape" style="top:73%;left:76%">
 <canvas id="triangle" class="shape" width="100"
height="50"></canvas>
 </div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Dynamic tooltip content in EJ2 JavaScript Tooltip control

The tooltip content can be changed dynamically using the [AJAX](#) request.

The AJAX request should be made within the [beforeRender](#) event of the tooltip. On every success, the corresponding retrieved data will be set to the [content](#) property of the tooltip.

When you hover over the icons, its respective data will be retrieved dynamically and then assigned to the tooltip's content.

Refer to the following code snippet to change the tooltip content dynamically.

```

`ts
function onBeforeRender(args: TooltipEventArgs): void {
 this.content = 'Loading...';
 this.dataBind();
 let ajax: Ajax = new Ajax('./tooltip.json', 'GET', true);
 ajax.send().then(
 (result: any) => {
 result = JSON.parse(result);
 for (let i: number = 0; i < result.length; i++) {
 if (result[i].Id == args.target.id) {

```

```

/ tslint:disable /
this.content = result[i].Sports;
/ tslint:enable /
}
}
this.dataBind();
},
(reason: any) => {
this.content = reason.message;
this.dataBind();
});
}
`

```

## INDEX.TS

```

import { Tooltip, TooltipEventArgs } from '@syncfusion/ej2-popups';
import { Ajax } from '@syncfusion/ej2-base';
let tooltip: Tooltip = new Tooltip({
 content: 'Loading...',
 target: '.circletool',
 showTipPointer: false,
 beforeRender: onBeforeRender
});
tooltip.appendTo('#box');
function onBeforeRender(args: TooltipEventArgs): void {
 this.content = 'Loading...';
 this.dataBind();
 let ajax: Ajax = new Ajax('./tooltip.json', 'GET', true);
 ajax.send().then(
 (result: any) => {
 result = JSON.parse(result);
 for (let i: number = 0; i < result.length; i++) {
 if (result[i].Id == args.target.id) {
 /* tslint:disable */
 this.content = result[i].Name;
 /* tslint:enable */
 }
 }
 this.dataBind();
 },
 (reason: any) => {
 this.content = reason.message;
 this.dataBind();
 }
);
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <h2> Dynamic Tooltip content </h2>
 <div id="box" class="e-prevent-select">
 <div id="1" class="circletool bold-01" style="display:inline-
block"></div>
 <div id="2" class="circletool italic" style="display:inline-
block"></div>
 <div id="3" class="circletool underline-02"
style="display:inline-block"></div>
 <div id="4" class="circletool cut-02" style="display:inline-
block"></div>
 <div id="5" class="circletool copy" style="display:inline-
block"></div>
 <div id="6" class="circletool paste" style="display:inline-
block"></div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

### Create and show tooltip on multiple targets in EJ2 JavaScript Tooltip control

Tooltip can be created and shown on multiple targets within a container by defining the specific target elements to the target property. So, the tooltip is initialized only on matched targets within a container.

In this case, the tooltip content is assigned from the title attribute of the target element.

`ts

```
import { Tooltip } from '@syncfusion/ej2-popups';

let tooltip: Tooltip = new Tooltip({
 position: 'RightCenter'
});

tooltip.appendTo('#uname'); // Here we have appended the target to the element.
```

## INDEX.TS

```
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
import { InputObject } from '@syncfusion/ej2-inputs';
let button1: Button = new Button();
button1.appendTo('#sample');
let button2: Button = new Button();
button2.appendTo('#clear');
let tooltip1: Tooltip = new Tooltip({
 position: 'RightCenter'
});
tooltip1.appendTo('#uname');
let tooltip2: Tooltip = new Tooltip({
 position: 'RightCenter'
});
tooltip2.appendTo('#mail');
let tooltip3: Tooltip = new Tooltip({
 position: 'RightCenter'
});
tooltip3.appendTo('#pwd');
let tooltip4: Tooltip = new Tooltip({
 position: 'RightCenter'
});
tooltip4.appendTo('#cpwd');
document.getElementById('sample').addEventListener('click', function() {
 let name = document.getElementById('uname').value;
 let pwd = document.getElementById('pwd').value;
 let cpwd = document.getElementById('cpwd').value;
 if(name.length > 0 & name.length < 4){
 document.getElementById('uname').title = 'Required Minimum 4
Characters';
 document.getElementById('uname').style.backgroundColor = 'red';
 let target = document.getElementById('uname');
 tooltip1.open(target);
 } else {
 document.getElementById('uname').style.backgroundColor = 'white';
 }
 if(pwd !== '' && pwd.length < 8){
 document.getElementById('pwd').title = 'Required Minimum 8 Characters';
 document.getElementById('pwd').style.backgroundColor = 'red';
 let pdwtargt = document.getElementById('pwd');
 tooltip3.open(pdwtargt);
 } else {
 document.getElementById('pwd').style.backgroundColor = 'white';
 }
 if(name.length >= 4 && pwd !== '' && pwd.length >= 8 && pwd == cpwd){
```



```

 alert('Form Submitted');
 } else {
 alert('Details are not Valid');
 }
})
document.getElementById('clear').addEventListener('click', function(){
 document.getElementById('uname').style.backgroundColor = 'white';
 document.getElementById('pwd').style.backgroundColor = 'white';
 let target = document.getElementById('uname');
 tooltip1.close(target);
 document.getElementById('uname').title = 'Please enter your name';
 let pwdtarget = document.getElementById('pwd');
 tooltip3.close(pwdtarget);
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tooltip</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript UI Controls">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <form id="details" role="form">
 <div id="user">
 <div class="info">User Name:</div>
 <div class="inputs"><input type="text" id="uname" class="e-info e-
input" name="firstname" title="Please enter your name"></div>
 </div>

 <div>
 <div class="info">Email Address:</div>
 <div class="inputs"><input type="text" id="mail" class="e-info e-
input" name="email" title="Enter a valid email address"></div>
 </div>

 <div>

```

```

 <div class="info">Password:</div>
 <div class="inputs"><input id="pwd" type="password" class="e-info
e-input" name="password" title="Be at least 8 characters length"></div>
 </div>

 <div>
 <div class="info">Confirm Password:</div>
 <div class="inputs"><input id="cpwd" type="password" class="e-info
e-input" name="Cpwd" title="Re-enter your password"></div>
 </div>

 <div class="btn">
 <input id="sample" type="button" value="Submit">
 <input id="clear" type="reset" value="Reset">
 </div>
</form>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## TreeGrid

### Getting started in EJ2 JavaScript Treegrid control

This section explains the steps required to create a simple Essential JS 2 TreeGrid and demonstrates the basic usage of the TreeGrid control in a JavaScript application.

### Dependencies

Following is the list of dependencies to use the TreeGrid with all features.

```
`javascript
```

```

|-- @syncfusion/ej2-treegrid
|-- @syncfusion/ej2-treegrid
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-splitbuttons

```

```
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-excel-export
|-- @syncfusion/ej2-pdf-export
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-compression
\
```

### Setup for local environment

Refer to the following steps to setup your local environment.

**Step 1:** Create a root folder `myapp` for your application.

**Step 2:** Create `myapp/resources` folder to store local scripts and styles files.

**Step 3:** Create `myapp/index.js` and `myapp/index.html` files for initializing Essential JS 2 TreeGrid control.

### Adding Syncfusion resources

The Essential JS 2 TreeGrid control can be initialized by using either of the following ways.

- Using local script and style.
- Using CDN link for script and style.

### Using local script and style

You can get the global scripts and styles from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, copy the TreeGrid and its dependencies scripts and style file into the `resources/scripts` and `resources/styles` folder.

Refer to the following code to find location TreeGrid's script and style file.

### Syntax:

Script: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js`

Styles: `(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css`

### Example:

Script: `C:/Program Files (x86)/Syncfusion/Essential Studio/16.4.40/Essential JS 2/ej2-treegrid/dist/global/ej2-treegrid.min.js`

Styles: `C:/Program Files (x86)/Syncfusion/Essential Studio/16.4.40/Essential JS 2/ej2-treegrid/styles/material.css`

After copying the files, refer the TreeGrid's scripts and styles into the `index.html` file.

The following HTML code example shows the minimal dependency of TreeGrid.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 TreeGrid</title>
<!-- Essential JS 2 TreeGrid's dependent material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="resources/treegrid/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 TreeGrid's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-grids.min.js" type="text/javascript"></script>
<!-- Essential JS 2 TreeGrid's global script -->
<script src="resources/scripts/ej2-treegrid.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
```

#### *Using CDN link for script and style*

Using CDN link, you can directly refer the TreeGrid control's script and style into the `index.html`.

Refer to the TreeGrid's CDN links as follows.

#### **Syntax:**

Script: `http://cdn.syncfusion.com/ej2/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

#### **Example:**

Script: <http://cdn.syncfusion.com/ej2/ej2-treegrid/dist/global/ej2-treegrid.min.js>

Styles: <http://cdn.syncfusion.com/ej2/ej2-treegrid/styles/material.css>

The following HTML code example shows the minimal dependency of TreeGrid.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 TreeGrid</title>
<!-- Essential JS 2 TreeGrid's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-treegrid/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 TreeGrid's dependent script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/ej2/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 TreeGrid's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-treegrid/dist/global/ej2-treegrid.min.js"
type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

### Adding TreeGrid control

Now, you can start adding TreeGrid control in the application. For getting started, add a `div` element for TreeGrid control in `index.html`. Then refer the `index.js` file into the `index.html` file.

In this document context, the `ej2.min.js` is used, which includes all the Essential JS 2 components and its dependent scripts.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 TreeGrid</title>
<!-- Essential JS 2 TreeGrid's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-treegrid/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 all script -->
<script src="http://cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for TreeGrid -->
<div id="TreeGrid"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
`
```

Place the following TreeGrid code in the `index.js`.

```
`javascript
var treeGridObj = ej.treegrid.TreeGrid();
treeGridObj.appendTo('#TreeGrid');
`
```

### Defining Row Data

Data for the TreeGrid control is bind by using the `dataSource` property. It accepts either an array of JavaScript object or `DataManager` instance.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 TreeGrid</title>
<!-- Essential JS 2 TreeGrid's dependent material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="http://cdn.syncfusion.com/ej2/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 material theme -->
<link href="http://cdn.syncfusion.com/ej2/ej2-treegrid/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 all script -->
<script src="http://cdn.syncfusion.com/ej2/dist/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element for TreeGrid -->
<div id="TreeGrid"></div>
<script src="index.js" type="text/javascript"></script>
</body>
</html>
`
```

Place the following TreeGrid code in the `index.js`.

```
`javascript
var sampleData = [
{
taskID: 1,
taskName: 'Planning',
startDate: new Date('02/03/2017'),
```

```

duration: 5,
subtasks: [
{ taskID: 2, taskName: 'Plan timeline', startDate: new Date('02/03/2017'), duration: 5 },
{ taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2017'), duration: 5},
{ taskID: 4, taskName: 'Allocate resources', startDate: new Date('02/03/2017'), duration: 5},
{ taskID: 5, taskName: 'Planning complete', startDate: new Date('02/07/2017'), duration: 0}
]
});
var treeGridObj = ej.treegrid.TreeGrid({dataSource: sampleData});
treeGridObj.appendTo('#TreeGrid');
`

```

### Defining Columns

The TreeGrid has an option to define the columns as an array. In these columns, the following properties are used to customize the columns.

- The `field` has been added to map with a property name in an array of JavaScript objects.
- The `headerText` has been added to change the title of columns.
- The `textAlign` has been added to change the alignment of columns. By default, columns will be left aligned. To change the columns to right align, define `textAlign` to `Right`.
- Also, the another useful property, `format` has been used. Using this, you can format number and date values to standard or custom formats. Here, it is defined for the conversion of numeric values to currency.

Tree Column which is used to expand or collapse child rows is defined by using [treeColumnIndex](#) property.

```

`javascript
var sampleData = [
{
taskID: 1,
taskName: 'Planning',
startDate: new Date('02/03/2017'),
duration: 5,
subtasks: [
{ taskID: 2, taskName: 'Plan timeline', startDate: new Date('02/03/2017'), duration: 5 },
{ taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2017'), duration: 5},
{ taskID: 4, taskName: 'Allocate resources', startDate: new Date('02/03/2017'), duration: 5},
{ taskID: 5, taskName: 'Planning complete', startDate: new Date('02/07/2017'), duration: 0}
]
}
]
`

```



```

]
});
var treeGridObj = ej.treegrid.TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200, textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90, textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign: 'Right' }
],
 height: 380
});
treeGridObj.appendTo('#TreeGrid');

```

In the above code example, the hierarchical data binding is represented in which the [childMapping](#) property denotes the hierarchy relationship; whereas in self-referencing data binding [idMapping](#) and [parentIdMapping](#) denotes the hierarchy relationship.

### Module injection

To create a treegrid with additional features, inject the required modules. The following modules are used to extend TreeGrid's basic functionality.

- **Page**: Inject this module to use paging feature.
- **Sort**: Inject this module to use sorting feature.
- **Filter**: Inject this module to use filtering feature.
- **ExcelExport**: Inject this module to use Excel export feature.
- **PdfExport**: Inject this module to use PDF export feature.

These modules should be injected into the TreeGrid using the `ej.treegrid.TreeGrid.Inject` method.

Additional feature modules are available [here](#).

### Enable paging

The paging feature enables users to view the TreeGrid record in a paged view. It can be enabled by setting the [allowPaging](#) property to true. Inject the [Page](#)

module as follows. If the [Page](#) module is not injected, the pager will not be rendered in the TreeGrid. The pager can be customized using the [pageSettings](#) property.

In root-level paging mode, paging is based on the root-level rows only, i.e., it ignores the child row count and it can be enabled by using the [pageSettings.pageSizeMode](#) property.

### INDEX.JS

```
ej.treegrid.TreeGrid.Inject(ej.treegrid.Page);
var treeGridObj = new ej.treegrid.TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 allowPaging: true,
 pageSettings: { pageSize: 7 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
 textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Enable sorting

The sorting feature enables you to order the records. It can be enabled by setting the [allowSorting](#) property to true. Inject the [Sort](#)

module as follows. If the [Sort](#) module is not injected, you cannot sort when a header is clicked. Sorting feature can be customized using the [sortSettings](#) property.

### INDEX.JS

```

ej.treegrid.TreeGrid.Inject(ej.treegrid.Page, ej.treegrid.Sort);
var treeGridObj = new ej.treegrid.TreeGrid({
 dataSource: sortData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 allowSorting: true,
 sortSettings: { columns: [{ field: 'Category', direction: 'Ascending' },
 { field: 'orderName', direction: 'Ascending' }] },
 allowPaging: true,
 columns: [
 { field: 'orderName', headerText: 'Order Name', width: 170
 },
 { field: 'Category', headerText: 'Category', width: 150 },
 { field: 'orderDate', headerText: 'Order Date', width: 130,
 textAlign: 'Right', format: 'yMd', type: 'date' },

```

```

 { field: 'price', headerText: 'Price', width: 100, format:
 'C0', textAlign: 'Right' }
],
 height: 260
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Enable filtering

The filtering feature enables you to view the reduced amount of records based on the filter criteria. It can be enabled by setting the [allowFiltering](#) property to true. The [Filter] module has to be injected as follows.

If the [Filter] module is not injected, filter bar will not be rendered in the TreeGrid. Filtering feature can be customized using the [filterSettings](#) property.

By default, filtered records are shown along with its parent records. This behavior can be changed by using the [filterSettings-hierarchyMode](#) property.

### INDEX.JS

```
ej.treegrid.TreeGrid.Inject(ej.treegrid.Page, ej.treegrid.Sort,
ej.grid.Filter);
var treeGridObj = new ej.treegrid.TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 allowFiltering: true,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
 textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
],
 pageSettings: {pageSize: 11},
 allowPaging: true,
 allowSorting: true
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Run the application

Now, run the `index.html` in the web browser, it will render the Essential JS 2 TreeGrid control.

Output will be displayed as follows.

### INDEX.JS

```

ej.treegrid.TreeGrid.Inject(ej.treegrid.Page, ej.treegrid.Sort,
ej.grid.Filter);
var treeGridObj = new ej.treegrid.TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 allowFiltering: true,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
],
 pageSettings: {pageSize: 11},
 allowPaging: true,
 allowSorting: true
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Deploy the application

The Essential JS 2 TreeGrid control features are segregated into individual feature-wise modules. The [Essential Studio JavaScript \(Essential JS 2\)](#) build and [CDN](#) scripts contains code for all features used in the TreeGrid. So, do not to use the [CDN](#) scripts in production. It is strongly recommended to generate script files to use in production using our **Custom Resource Generator(CRG)** for Essential JS 2. CRG allows you to generate the bundled script for the currently enabled features in TreeGrid.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to knows how to present and manipulate data.

### Module in EJ2 JavaScript Treegrid control

The following feature modules should be injected to extend the TreeGrid's functionality.

Module	Description
----- -----	
<a href="#">PageService</a>	Inject this module to use paging feature.
<a href="#">SortService</a>	Inject this module to use sorting feature.
<a href="#">FilterService</a>	Inject this module to use filtering feature.
<a href="#">EditService</a>	Inject this module to use editing feature.
<a href="#">AggregateService</a>	Inject this module to use aggregate feature.
<a href="#">ColumnChooserService</a>	Inject this module to use column chooser feature.



- | [ColumnMenuService](#) | Inject this module to use column menu feature. |
- | [CommandColumnService](#) | Inject this module to use command column feature. |
- | [ContextMenuService](#) | Inject this module to use context menu feature.
- | [ResizeService](#) | Inject this module to use resize feature. |
- | [ReorderService](#) | Inject this module to use reorder feature. |
- | [PrintService](#) | Inject this module to use to use print feature and this is a default injected module. |
- | [ToolbarService](#) | Inject this module to use toolbar feature. |
- | [ExcelExportService](#) | Inject this module to use Excel export feature. |
- | [PdfExportService](#) | Inject this module to use PDF export feature. |

These modules should be injected into the TreeGrid using the `ej.treegrid..TreeGrid.Inject` method.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

## Data Binding

### Data binding in EJ2 JavaScript Treegrid control

The TreeGrid uses `DataManager`, which supports both RESTful JSON data services binding and local JavaScript object array binding. The `dataSource` property can be assigned either with the instance of `DataManager` or JavaScript object array collection.

It supports two kinds of data binding method:

- Local data
- Remote data

### Binding with ajax

You can use TreeGrid `dataSource` property to bind the data source to TreeGrid from external Fetch request. In the below code we have fetched the data source from the server with the help of Ajax request and provided that to `dataSource` property by using `onSuccess` event of the Fetch.

### INDEX.TS

```
import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { Fetch } from '@syncfusion/ej2-base';
TreeGrid.Inject(Page);
let treegrid: TreeGrid = new TreeGrid({
 idMapping: 'TaskID',
 parentIdMapping: 'ParentItem',
 pageSettings: { pageSize: 7 },
 allowPaging: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
```

```

 { field: 'Progress', headerText: 'Progress', width: 80, textAlign:
 'Right' }
]
});
treegrid.appendTo('#TreeGrid');
let button: HTMLElement = document.createElement('button');
button.textContent = 'Bind Data';
treegrid.element.parentNode.insertBefore(button, treegrid.element);
button.addEventListener("click", function(e) {
 let fetch = new
Fetch("https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData", "GET");
 treegrid.showSpinner();
 fetch.send();
 fetch.onSuccess = function (data: object) {
 treegrid.hideSpinner();
 treegrid.dataSource = data;
 };
});
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

#### *Handling expandStateMapping*

To denotes the expand status of parent row, define the [expandStateMapping](#) property of tree grid.

The `expandStateMapping` property maps the field name in data source, that denotes whether parent record is in expanded or collapsed state and this is useful to renders parent row in expanded or collapsed state based on this mapping property value in data source.

`ts

```

import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { TreeGridComponent } from '@syncfusion/ej2-treegrid';
import './App.css';

let dataManager: DataManager = new DataManager({
 adaptor: new UrlAdaptor,
 url: "Home/DataSource",
});

let treegrid: TreeGrid = new TreeGrid({
 dataSource: dataManager,
 hasChildMapping : 'isParent',
 idMapping: 'TaskID',
 expandStateMapping: 'IsExpanded',
 parentIdMapping: 'ParentValue',

```

```

height: 400,
treeColumnIndex: 1,
columns: [
{ field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width: 90 },
{ field: 'TaskName', headerText: 'Task Name', width: 180 },
{ field: 'Duration', headerText: 'Duration', width: 80, textAlign: 'Right' }
]
});
treegrid.appendTo('#TreeGrid');
`

```

The following code example defines `expandStateMapping` property at server end.

```

`ts
public ActionResult ExpandStateMapping()
{
return View();
}

public class TreeData
{
public static List<TreeData> tree = new List<TreeData>();
[System.ComponentModel.DataAnnotations.Key]
public int TaskID { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int? ParentValue { get; set; }
public bool? isParent { get; set; }
public bool IsExpanded { get; set; }
public TreeData() { }
public static List<TreeData> GetTree()
{
if (tree.Count == 0)
{
int root = 0;
for (var t = 1; t <= 500; t++)

```

```

{
 Random ran = new Random();
 string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
 string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
 root++;
 int rootItem = root;
 tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent
 = true, IsExpanded = false, ParentValue = null, Duration = ran.Next(1, 50) });
 int parent = root;
 for (var d = 0; d < 1; d++)
 {
 root++;
 string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";
 int par = parent + 1;
 progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
 int iD = root;
 tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
 IsExpanded = false, ParentValue = rootItem, Duration = ran.Next(1, 50) });
 int subparent = root;
 for (var c = 0; c < 500; c++)
 {
 root++;
 string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";
 int subchild = subparent + c + 1;
 string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
 int childID = root ;
 tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(), isParent =
 false, IsExpanded = false, ParentValue = subparent, Duration = ran.Next(1, 50) });
 }
 }
}
return tree;
}

```

```
}
,
```

### Custom binding

It is possible to handle data processing externally and bind the result to the TreeGrid. This helps you to provide your own custom data logic. TreeGrid expects an object as the result of the custom logic and the emitted value should be an object with properties result and count.

In this context, we are going to use DataManager with WebApi Adaptor for handling remote interaction, you can choose any HTTP client as per your choice.

```
`ts
```

```
import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor, Query } from '@syncfusion/ej2-data';
import { isNullOrUndefined } from '@syncfusion/ej2-base';

TreeGrid.Inject(Page);

let data: DataManager = new DataManager({
 url: 'http://localhost:51473/api/Tasks', // url for fetching the data from server
 adaptor: new WebApiAdaptor,
 crossDomain: true
});

///// filter query for fetching the root level records only and take value should be equal to the
pageSize value of pageSettings

data.executeQuery(new Query().where('ParentId', 'equal', null).take(3).skip(0).requiresCount()).then((e:
any) => {

 let treegrid: TreeGrid = new TreeGrid({
 dataSource: { result: e.result, count: e.count },
 hasChildMapping: 'IsParent',
 idMapping: 'TaskId',
 parentIdMapping: 'ParentId',
 allowPaging: true,
 pageSettings: { pageSize: 3, pageSizeMode: 'Root' },
 dataStateChange: function (state: any) {
 if (state.action.requestType === 'paging') {
 data.executeQuery(new Query().skip(state.skip).take(state.take).requiresCount()).then(function (e: any)
 {
 this.dataSource = { result: e.result, count: e.count };
 });
 }
 }
 });
```

```

 }
 },
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskId', headerText: 'Task ID', isPrimaryKey: true, textAlign: 'Right', width: 120 },
 { field: 'TaskName', headerText: 'Task Name', width: 150 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 120, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 110, textAlign: 'Right' },
 { field: 'Progress', headerText: 'Progress', width: 110 },
]
});
treegrid.appendTo('#TreeGrid');
});

```

We have a limitation for Custom Binding feature of TreeGrid. This feature works only for Self Referential data binding with `pageSizeMode` as `Root`.

#### Handling child data

Using the custom binding feature you can bind the child data for a parent record as per your custom logic. When a parent record is expanded, [dataStateChange](#) event is triggered in which you can assign your custom data to the `childData` property of the [dataStateChange](#) event arguments.

After assigning the child data, `childDataBind` method should be called from the [dataStateChange](#) event arguments to indicate that the data is bound.

In this context, initially we have assigned only the parent records to the treegrid `dataSource` and fetched the required child records in the [dataStateChange](#) event.

ts

```

import { TreeGrid, Page, Toolbar, Sort } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor, Query } from '@syncfusion/ej2-data';
import { isNullOrUndefined } from '@syncfusion/ej2-base';

TreeGrid.Inject(Page);

let data: DataManager = new DataManager({
 url: 'http://localhost:51473/api/Tasks', // url for fetching the data from server
 adaptor: new WebApiAdaptor,
 crossDomain: true
});

```

```
/// filter query for fetching only the root level records
data.executeQuery(new Query().where('ParentId', 'equal', null).take(3).skip(0).requiresCount()).then((e: any) => {
let treegrid: TreeGrid = new TreeGrid({
 dataSource: { result: e.result, count: e.count },
 hasChildMapping: 'IsParent',
 idMapping: 'TaskId',
 parentIdMapping: 'ParentId',
 allowPaging: true,
 pageSettings: { pageSize: 3, pageSizeMode: 'Root' },
 dataStateChange: function (state: any) {
 if (!NullOrUndefined(state.requestType) && state.requestType === 'expand') {
 /// filter query for fetching the child records of the expanded record
 data.executeQuery(new Query().where('ParentId', 'equal', state.data.TaskId)).then(function (e: any) {
 state.childData = e.result;
 state.childDataBind();
 });
 } else {
 data.executeQuery(new Query().skip(state.skip).take(state.take).requiresCount()).then(function (e: any) {
 {
 this.dataSource = { result: e.result, count: e.count };
 }
 });
 }
 },
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskId', headerText: 'Task ID', isPrimaryKey: true, textAlign: 'Right', width: 120 },
 { field: 'TaskName', headerText: 'Task Name', width: 150 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 120, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 110, textAlign: 'Right' },
 { field: 'Progress', headerText: 'Progress', width: 110 },
]
});
```



```
treegrid.appendTo('#TreeGrid');
});
```

### Handling Tree Grid actions

For TreeGrid actions such as paging, sorting, etc dataStateChange event will be invoked. You have to query and resolve data using Fetch in this event based on the state arguments.

```
`ts
import { TreeGrid, Page, Toolbar, Sort } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor, Query } from '@syncfusion/ej2-data';
import { isNullOrUndefined } from '@syncfusion/ej2-base';

TreeGrid.Inject(Page);

let data: DataManager = new DataManager({
 url: 'http://localhost:51473/api/Tasks', /// url for fetching the data from server
 adaptor: new WebApiAdaptor,
 crossDomain: true
});

/// filter query for fetching only the root level records
data.executeQuery(new Query().where('ParentId', 'equal', null).take(3).skip(0).requiresCount()).then((e: any) => {
 let treegrid: TreeGrid = new TreeGrid({
 dataSource: { result: e.result, count: e.count },
 hasChildMapping: 'IsParent',
 idMapping: 'TaskId',
 parentIdMapping: 'ParentId',
 allowPaging: true,
 allowSorting: true,
 pageSettings: { pageSize: 3, pageSizeMode: 'Root' },
 dataStateChange: function (state: any) {
 if (!isNullOrUndefined(state.requestType) && state.requestType === 'expand') {
 /// filter query for fetching the child records of the expanded record
 data.executeQuery(new Query().where('ParentId', 'equal', state.data.TaskId)).then(function (e: any) {
 state.childData = e.result;
 state.childDataBind();
 });
 }
 }
 });
```

```

}
if (state.action.requestType === 'paging') {
 data.executeQuery(new Query().skip(state.skip).take(state.take).requiresCount()).then(function (e: any)
 {
 this.dataSource = { result: e.result, count: e.count };
 });
}
if (state.action.requestType === 'sorting') {
 //// sort query for getting the sorted records from server
 data.executeQuery(new Query().sortBy('TaskId',
 'descending').addParams('IdMapping','TaskId').skip(state.skip).take(state.take).requiresCount()).then(fu
 nction (e: any) {
 this.dataSource = { result: e.result, count: e.count };
 });
}
},
treeColumnIndex: 1,
columns: [
 { field: 'TaskId', headerText: 'Task ID', isPrimaryKey: true, textAlign: 'Right', width: 120 },
 { field: 'TaskName', headerText: 'Task Name', width: 150 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 120, format: { skeleton: 'yMd', type:
 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 110, textAlign: 'Right' },
 { field: 'Progress', headerText: 'Progress', width: 110 },
]
});
treegrid.appendTo('#TreeGrid');
});
`

```

#### Performing CRUD actions

The [dataSourceChanged](#) event will be triggered for updating the treegrid data. You can perform the save operation based on the event arguments and call the endEdit method to indicate the completion of save operation.

ts

```
import { TreeGrid, Page, Toolbar, Sort, Filter, Edit } from '@syncfusion/ej2-treegrid';
```

```
import { DataManager, WebApiAdaptor, Query } from '@syncfusion/ej2-data';
import { isNullOrUndefined } from '@syncfusion/ej2-base';

TreeGrid.Inject(Page, Toolbar, Sort, Filter, Edit);

let data: DataManager = new DataManager({
 url: 'http://localhost:51473/api/Tasks',
 adaptor: new WebApiAdaptor,
 crossDomain: true
});

data.executeQuery(new Query().where('ParentId', 'equal', null).take(3).skip(0).requiresCount()).then((e: any) => {

 let treegrid: TreeGrid = new TreeGrid({
 dataSource: { result: e.result, count: e.count },
 hasChildMapping: 'IsParent',
 idMapping: 'TaskId',
 parentIdMapping: 'ParentId',
 allowPaging: true,
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Row' },
 pageSettings: { pageSize: 3, pageSizeMode: 'Root' },
 dataSourceChanged: function (state: any) {
 if (state.action == 'add') {
 state.endEdit();
 }
 else if (state.action == 'edit') {
 state.endEdit();
 }
 else if (state.requestType == 'delete') {
 state.endEdit();
 }
 },
 dataStateChange: function (state: any) {
 if (!isNullOrUndefined(state.requestType) && state.requestType === 'expand') {
 data.executeQuery(new Query().where('ParentId', 'equal', state.data.TaskId)).then(function (e: any) {
```

```

state.childData = e.result;
state.childDataBind();
});
} else {
data.executeQuery(new Query().where('ParentId', 'equal',
null).take(3).skip(0).requiresCount()).then(function (e: any) {
this.dataSource = { result: e.result, count: e.count };
});
}
},
treeColumnIndex: 1,
columns: [
{ field: 'TaskId', headerText: 'Task ID', isPrimaryKey: true, textAlign: 'Right', width: 120 },
{ field: 'TaskName', headerText: 'Task Name', width: 150 },
{ field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 120, format: { skeleton: 'yMd', type:
'date' } },
{ field: 'Duration', headerText: 'Duration', width: 110, textAlign: 'Right' },
{ field: 'Progress', headerText: 'Progress', width: 110 },
]
});
treegrid.appendTo('#TreeGrid');
});

```

#### Calculate aggregates

The footer aggregate values should be calculated and sent along with the **dataSource** property as follows. The aggregate property of the data source should contain the aggregate value assigned to the property named in the **field – type** format. For example, the **Sum** aggregate value for the **Duration** field should be assigned to the property named as **Duration - sum**.

```

、
{
result: [{..}, {..}, {..}, ...],
count: 830,
aggregates: { 'Freight - sum' : 450,'EmployeeID - min': 1 }
}
、

```

### Provide excel filter data source

The [dataStateChange](#) event will be triggered with appropriate arguments when the excel filter requests the filter choice data source. You need to resolve the excel filter data source using the **dataSource** resolver function from the state argument as follows.

```
`ts
import { TreeGrid, Page, Filter } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor, Query } from '@syncfusion/ej2-data';
import { isNullOrUndefined } from '@syncfusion/ej2-base';

TreeGrid.Inject(Page, Filter);

let data: DataManager = new DataManager({
 url: 'http://localhost:51473/api/Tasks', //// url for fetching the data from server
 adaptor: new WebApiAdaptor,
 crossDomain: true
});

///// filter query for fetching the root level records only
data.executeQuery(new Query().where('ParentId', 'equal', null).take(3).skip(0).requiresCount()).then((e: any) => {

 let treegrid: TreeGrid = new TreeGrid({
 dataSource: { result: e.result, count: e.count },
 hasChildMapping: 'IsParent',
 idMapping: 'TaskId',
 parentIdMapping: 'ParentId',
 allowPaging: true,
 allowFiltering: true,
 filterSettings: { type: 'Excel' },
 pageSettings: { pageSize: 3, pageSizeMode: 'Root' },
 dataStateChange: function (state: any) {
 if (state.action && (state.action.requestType === 'filterchoicerequest'
 || state.action.requestType === 'filtersearchbegin')) {
 data.executeQuery(new Query().skip(state.skip).take(state.take).requiresCount()).then(function (e: any) {
 state.dataSource && state.dataSource(e); /// resolve the excel filter dataSource
 });
 }
 }
 });
 else {
```

```

/// take value must be always be equal to the pageSize value of pageSetings
data.executeQuery(new Query().skip(state.skip).take(state.take).requiresCount()).then(function (e: any)
{
this.dataSource = { result: e.result, count: e.count };
});
}
},
treeColumnIndex: 1,
columns: [
{ field: 'TaskId', headerText: 'Task ID', isPrimaryKey: true, textAlign: 'Right', width: 120 },
{ field: 'TaskName', headerText: 'Task Name', width: 150 },
{ field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 120, format: { skeleton: 'yMd', type: 'date' } },
{ field: 'Duration', headerText: 'Duration', width: 110, textAlign: 'Right' },
{ field: 'Progress', headerText: 'Progress', width: 110 },
]
});
treegrid.appendTo('#TreeGrid');
});
`

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Local data in EJ2 JavaScript Treegrid control

In Local Data binding, data source for rendering the TreeGrid control is retrieved from the same application locally.

Two types of Data binding are possible with the TreeGrid control.

- Hierarchical Datasource binding
- Self-Referential Data binding (Flat Data)

To bind local data to the treegrid, you can assign a JavaScript object array to the [dataSource](#) property. The local data source can also be provided as an instance of the `DataManager`.

By default, `DataManager` uses `JsonAdaptor` for local data-binding.

#### Hierarchy data source binding

The [childMapping](#) property is used to map the child records in hierarchy data source.

The following code example shows you how to bind the hierarchical local data into the TreeGrid control.

**INDEX.TS**

```
import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 pageSettings: {pageSize: 7},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* Remote data binding is not supported for Hierarchy Data.

#### Self-Referential data binding (Flat data)

TreeGrid is rendered from Self-Referential data structures by providing two fields, ID field and parent ID field.

- **ID Field:** This field contains unique values used to identify nodes. Its name is assigned to the [idMapping](#) property.
- **Parent ID Field:** This field contains values that indicate parent nodes. Its name is assigned to the [parentIdMapping](#) property.

#### INDEX.TS

```

import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { projectData } from './datasource.ts';
TreeGrid.Inject(Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: 'TaskID',
 parentIdMapping: 'parentID',
 allowPaging: true,
 treeColumnIndex: 1,
 pageSettings: { pageSize: 7 },
 columns: [
 { field: 'TaskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 {

```



```

 field: 'StartDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date',format: 'yMd'
 },
 { field: 'Duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">

```

```

 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Herewith we have provided list of reserved properties and the purpose we used internally in TreeGrid. We recommend to avoid these reserved properties in your dataSource (To get rid of conflicts).

#### Reserved keywords | Purpose

childRecords | Specifies the childRecords of a parentData

hasChildRecords | Specifies whether the record contains child records

hasFilteredChildRecords | Specifies whether the record contains filtered child records

expanded | Specifies whether the child records are expanded

parentItem | Specifies the parentItem of childRecords

index | Specifies the index of current record

level | Specifies the hierarchy level of record

filterLevel | Specifies the hierarchy level of filtered record

parentIdMapping | Specifies the parentId

uniqueID | Specifies the unique ID of a record

parentUniqueID | Specifies the parent Unique ID of a record

checkboxState | Specifies the checkbox state of a record

isSummaryRow | Specifies the summary of a record

taskData | Specifies the main data

primaryParent | Specifies the Primary data

#### Remote data in EJ2 JavaScript Treegrid control

To bind remote data to TreeGrid component, assign service data as an instance of **DataManager** to the [dataSource](#) property. To interact with remote data source, provide the endpoint [url](#) and define the [hasChildMapping](#) property of treegrid.

The [hasChildMapping](#) property maps the field name in data source, that denotes whether current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

The TreeGrid provides **Load on Demand** support for rendering remote data. The Load on demand is considered in TreeGrid for the following actions.

- Expanding root nodes.

- Navigating pages, with paging enabled in TreeGrid.

When load on demand is enabled, all the root nodes are rendered in collapsed state at initial load.

When load on demand support is enabled in TreeGrid with paging, the current or active page's root node alone will be rendered in collapsed state. On expanding the root node, the child nodes will be loaded from the remote server.

When a root node is expanded, its child nodes are rendered and are cached locally, such that on consecutive expand/collapse actions on root node, the child nodes are loaded from the cache instead from the remote server.

Similarly, if the user navigates to a new page, the root nodes of that specific page, will be rendered with request to the remote server.

Remote Data Binding supports only Self-Referential Data and by default the `pageSizeMode` for Remote Data is `Root` mode. i.e only root node's count will be shown in pager while using Remote Data

### INDEX.TS

```
import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
 url: 'https://ej2services.syncfusion.com/production/web-
 services/api/SelfReferenceData',
 adaptor: new WebApiAdaptor, crossDomain: true
});
TreeGrid.Inject(Page);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: data,
 hasChildMapping: 'isParent',
 idMapping: 'TaskID',
 parentIdMapping: 'ParentItem',
 height: 260,
 allowPaging: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right',
width: 90, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treegrid.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, **DataManager** uses **ODataAdaptor** for remote data-binding.

Based on the RESTful web services, set the corresponding adaptor to **DataManager**. Refer [here](#) for more details.

Filtering and searching server-side data operations are not supported in load on demand

### *LoadChildOnDemand*

While binding remote data to Tree Grid component, by default Tree Grid renders parent rows in collapsed state. Tree Grid provides option to load the child records also during the initial rendering itself for remote data binding by setting [loadChildOnDemand](#) as true.

When [loadChildOnDemand](#) is enabled parent records are rendered in expanded state.

The following code example describes the behavior of the loadChildOnDemand feature of Tree Grid.

Also while using **loadChildOnDemand** we need to handle the child records on server end and it is applicable to CRUD operations also.

The following code example describes handling of child records at server end.

```
`ts
public ActionResult UrlDatasource(DataManagerRequest dm)
{
 if (TreeData.tree.Count == 0)
 TreeData.GetTree();
 IEnumerable DataSource = TreeData.tree;
 DataOperations operation = new DataOperations();
 if (dm.Where != null && dm.Where.Count > 0)
 {
 DataSource = operation.PerformFiltering(DataSource, dm.Where, "and"); //perform filtering
 }
 if (dm.Sorted != null && dm.Sorted.Count > 0)
 {
 DataSource = operation.PerformSorting(DataSource, dm.Sorted); //perform sorting
 }
 var count = DataSource.ToList<TreeData>().Count();
 if (dm.Skip != 0)
 {
 DataSource = operation.PerformSkip(DataSource, dm.Skip); //Paging
 }
 if (dm.Take != 0)
 {
 DataSource = operation.PerformTake(DataSource, dm.Take);
 }
 if (dm.Where != null)
```

```

{
DataSource = CollectChildRecords(DataSource, dm); // method to collect child records
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}

public IEnumerable CollectChildRecords(IEnumerable datasource, DataManagerRequest dm)
{
DataOperations operation = new DataOperations();
IEnumerable DataSource = TreeData.tree; // use the total DataSource here
string IdMapping = "TaskID"; // define your IdMapping field name here
int[] TaskIds = new int[0];
foreach (var rec in datasource)
{
int taskid = (int)rec.GetType().GetProperty(IdMapping).GetValue(rec);
TaskIds = TaskIds.Concat(new int[] { taskid }).ToArray(); //get the Parentrecord Ids based on
IdMapping Field
}
IEnumerable ChildRecords = null;
foreach (int id in TaskIds)
{
dm.Where[0].value = id;
IEnumerable records = operation.PerformFiltering(DataSource, dm.Where, dm.Where[0].Operator);
//perform filtering to collect the childrecords based on Ids
ChildRecords = ChildRecords == null || (ChildRecords.AsQueryable().Count() == 0) ? records :
((IEnumerable<object>)ChildRecords).Concat((IEnumerable<object>)records); //concat the
childrecords with dataSource
}
if (ChildRecords != null)
{
ChildRecords = CollectChildRecords(ChildRecords, dm); // repeat the operation for inner level child
if (dm.Sorted != null && dm.Sorted.Count > 0) // perform Sorting
{
ChildRecords = operation.PerformSorting(ChildRecords, dm.Sorted);
}
}
}

```

```

datasource = ((IEnumerable<object>)datasource).Concat((IEnumerable<object>)ChildRecords);
//concat the childrecords with dataSource
}

return datasource;
}
`

```

### Offline mode

On remote data binding, all treegrid actions such as paging, loading child on-demand, will be processed on server-side. To avoid postback, set the treegrid to load all data on initialization and make the actions process in client-side. To enable this behavior, use the `offline` property of `DataManager`.

### INDEX.TS

```

import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
 url: 'https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData',
 adaptor: new WebApiAdaptor, crossDomain: true, offline: true
});
TreeGrid.Inject(Page);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: data,
 idMapping: 'TaskID',
 parentIdMapping: 'ParentItem',
 height: 260,
 allowPaging: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right',
width: 90, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treegrid.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Custom adaptor

You can create your own adaptor by extending the built-in adaptors. The following demonstrates custom adaptor approach and how to add a serial number for the records by overriding the built-in response processing using the `processResponse` method of the `WebApiAdaptor`.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';

```



```

class SerialNoAdaptor extends WebApiAdaptor {
 public processResponse(): Object[] {
 let i: number = 0;
 // calling base class processResponse function
 let original: Object[] = super.processResponse.apply(this,
arguments);
 // adding serial number
 original.forEach((item: Object) => item['Sno'] = ++i);
 return original;
 }
}
let data: DataManager = new DataManager({
 url: 'https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData',
 adaptor: new SerialNoAdaptor, crossDomain: true, offline: true
});
let treegrid: TreeGrid = new TreeGrid({
 dataSource: data,
 idMapping: 'TaskID',
 parentIdMapping: 'ParentItem',
 height: 260,
 treeColumnIndex: 1,
 columns: [
 { field: 'Sno', width: 120, headerText: 'SNO', textAlign: 'Right' },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right',
width: 90, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign:
'Reight' }
]
});
treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

#### *Sending additional parameters to the server*

To add a custom parameter to the data request, use the `addParams` method of `Query` class. Assign the `Query` object with additional parameters to the treegrid `query` property.

#### **INDEX.TS**

```

import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { DataManager, Query, WebApiAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
 url: 'https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData',
 adaptor: new WebApiAdaptor, crossDomain: true
});
TreeGrid.Inject(Page);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: data,
 hasChildMapping: 'isParent',
 idMapping: 'TaskID',
 parentIdMapping: 'ParentItem',
 height: 260,
 allowPaging: true,

```

```

 query: new Query().addParams('ej2treegrid', 'true'),
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right',
width: 90, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign:
'Reight' }
]
 });
 treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Handling HTTP error

During server interaction from the treegrid, some server-side exceptions may occur, and you can acquire those error messages or exception details

in client-side using the [actionFailure](#) event.

The argument passed to the [actionFailure](#) event contains the error details returned from the server.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { DataManager } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
 url: 'http://some.com/invalidUrl'
});
let treegrid: TreeGrid = new TreeGrid({
 dataSource: data,
 idMapping: 'TaskID',
 parentIdMapping: 'ParentItem',
 actionFailure: (e) => {
 let span: HTMLElement = document.createElement('span');
 treegrid.element.parentNode.insertBefore(span, treegrid.element);
 span.style.color = '#FF0000'
 span.innerHTML = 'Server exception: 404 Not found';
 },
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right',
width: 90, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treegrid.appendTo('#TreeGrid');

```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

The [actionFailure](#) event will be triggered not only for the server errors, but also when there is an exception while processing the treegrid actions.

#### *Load on demand with virtualization*

While binding remote data to Tree Grid component, by default Tree Grid renders parent rows in collapsed state. When expanding the root node, the child nodes will be loaded from the remote server.

When using virtualization with remote data binding, it helps you to improve the tree grid performance while loading a large set of data by setting [enableVirtualization](#) as true. The Tree Grid UI virtualization allows it to render only rows and columns visible within the view-port without buffering the entire datasource.

[hasChildMapping](#) property maps the field name in data source, that denotes whether current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

The following code example describes handling of Load on demand at server end.

```
`ts
public ActionResult lazyLoading()
{
 TreeData.tree = new List<TreeData>();
 return View();
}

public ActionResult UrlDatasource(DataManagerRequest dm)
{
 List<TreeData> data = new List<TreeData>();
 data = TreeData.GetTree();
 DataOperations operation = new DataOperations();
 IEnumerable<TreeData> DataSource = data;
 List<TreeData> ExpandedParentRecords = new List<TreeData>();
 if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the ExpandStateMapping
 property whether is true or false
 {
 var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
 val.IsExpanded = true;
 }
 else if (dm.Expand != null && dm.Expand[0] == "CollapsingAction")
 {
 var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
 val.IsExpanded = false;
 }
}
```

```
}
if (!(dm.Where != null && dm.Where.Count > 1))
{
 data = data.Where(p => p.ParentValue == null).ToList();
}
DataSource = data;
if (dm.Search != null && dm.Search.Count > 0) // Searching
{
 DataSource = operation.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting
{
 DataSource = operation.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 1) //filtering
{
 DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");
}
data = new List<TreeData>();
foreach (var rec in DataSource)
{
 if (rec.IsExpanded)
 {
 ExpandedParentRecords.Add(rec as TreeData); // saving the expanded parent records
 }
 data.Add(rec as TreeData);
}
var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.ParentValue)
 .Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
if (ExpandedParentRecords.Count > 0)
{
 foreach (var Record in ExpandedParentRecords.ToList())
 {
```

```
var ChildGroup = GroupData[Record.TaskID.ToString()];
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
{
 IEnumerable ChildSort = ChildGroup;
 ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
 ChildGroup = new List<TreeData>();
 foreach (var rec in ChildSort)
 {
 ChildGroup.Add(rec as TreeData);
 }
}
if (dm.Search != null && dm.Search.Count > 0) // searching the child records
{
 IEnumerable ChildSearch = ChildGroup;
 ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
 ChildGroup = new List<TreeData>();
 foreach (var rec in ChildSearch)
 {
 ChildGroup.Add(rec as TreeData);
 }
}
AppendChildren(dm, ChildGroup, Record, GroupData, data);
}
}

DataSource = data;
if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the skip index based on
collapsed parent
{
 string IdMapping = "TaskID";
 List<WhereFilter> CollapseFilter = new List<WhereFilter>();
 CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
 var CollapsedParentRecord = operation.PerformFiltering(DataSource, CollapseFilter, "and");
 var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
```



```

dm.Skip = index;
}

else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the skip index based on
expanded parent
{
 string IdMapping = "TaskID";
 List<WhereFilter> ExpandFilter = new List<WhereFilter>();
 ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
 var ExpandedParentRecord = operation.PerformFiltering(DataSource, ExpandFilter, "and");
 var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
 dm.Skip = index;
}

int count = data.Count;
DataSource = data;
if (dm.Skip != 0)
{
 DataSource = operation.PerformSkip(DataSource, dm.Skip); //Paging
}
if (dm.Take != 0)
{
 DataSource = operation.PerformTake(DataSource, dm.Take);
}

return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}

private void AppendChildren(DataManagerRequest dm, List<TreeData> ChildRecords, TreeData
ParentValue, Dictionary<string, List<TreeData>> GroupData, List<TreeData> data) // Getting child
records for the respective parent
{
 string TaskId = ParentValue.TaskID.ToString();
 var index = data.IndexOf(ParentValue);
 DataOperations operation = new DataOperations();
 foreach (var Child in ChildRecords)
 {

```

```
if (ParentValue.IsExpanded)
{
 string ParentId = Child.ParentValue.ToString();
 if (TaskId == ParentId)
 {
 ((IList)data).Insert(++index, Child);
 if (GroupData.ContainsKey(Child.TaskID.ToString()))
 {
 var DeepChildRecords = GroupData[Child.TaskID.ToString()];
 if (DeepChildRecords?.Count > 0)
 {
 if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
 {
 IEnumerable ChildSort = DeepChildRecords;
 ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
 DeepChildRecords = new List<TreeData>();
 foreach (var rec in ChildSort)
 {
 DeepChildRecords.Add(rec as TreeData);
 }
 }
 if (dm.Search != null && dm.Search.Count > 0) // searching the child records
 {
 IEnumerable ChildSearch = DeepChildRecords;
 ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
 DeepChildRecords = new List<TreeData>();
 foreach (var rec in ChildSearch)
 {
 DeepChildRecords.Add(rec as TreeData);
 }
 }
 AppendChildren(dm, DeepChildRecords, Child, GroupData, data);
 }
 if (Child.IsExpanded)
```

```
{
index += DeepChildRecords.Count;
}
}
else
{
Child.isParent = false;
}
}
}
}
}

public ActionResult Update(CRUDModel<TreeData> value)
{
List<TreeData> data = new List<TreeData>();
data = TreeData.GetTree();
var val = data.Where(ds => ds.TaskID == value.Value.TaskID).FirstOrDefault();
val.TaskName = value.Value.TaskName;
val.Duration = value.Value.Duration;
return Json(val);
}

public ActionResult Insert(CRUDModel<TreeData> value)
{
var c = 0;
for (; c < TreeData.GetTree().Count; c++)
{
if (TreeData.GetTree()[c].TaskID == value.RelationalKey)
{
if (TreeData.GetTree()[c].isParent == null)
{
TreeData.GetTree()[c].isParent = true;
}
}
```

```
break;
}
}
c += FindChildRecords(value.RelationalKey);
TreeData.GetTree().Insert(c + 1, value.Value);
return Json(value.Value);
}
public int FindChildRecords(int? id)
{
 var count = 0;
 for (var i = 0; i < TreeData.GetTree().Count; i++)
 {
 if (TreeData.GetTree()[i].ParentValue == id)
 {
 count++;
 count += FindChildRecords(TreeData.GetTree()[i].TaskID);
 }
 }
 return count;
}
public object Delete(CRUDModel<TreeData> value)
{
 if (value.deleted != null)
 {
 for (var i = 0; i < value.deleted.Count; i++)
 {
 TreeData.GetTree().Remove(TreeData.GetTree().Where(ds => ds.TaskID ==
 value.deleted[i].TaskID).FirstOrDefault());
 }
 }
 else
 {
 TreeData.GetTree().Remove(TreeData.GetTree().Where(or => or.TaskID ==
 int.Parse(value.Key.ToString())).FirstOrDefault());
 }
}
```

```

}
return Json(value);
}
public class CRUDModel<T> where T : class
{
 public TreeData Value;
 public int Key { get; set; }
 public int RelationalKey { get; set; }
 public List<T> added { get; set; }
 public List<T> changed { get; set; }
 public List<T> deleted { get; set; }
}
public class TreeData
{
 public static List<TreeData> tree = new List<TreeData>();
 [System.ComponentModel.DataAnnotations.Key]
 public int TaskID { get; set; }
 public string TaskName { get; set; }
 public int Duration { get; set; }
 public int? ParentValue { get; set; }
 public bool? isParent { get; set; }
 public bool IsExpanded { get; set; }
 public TreeData() { }
 public static List<TreeData> GetTree()
 {
 if (tree.Count == 0)
 {
 int root = 0;
 for (var t = 1; t <= 500; t++)
 {
 Random ran = new Random();
 string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
 string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
 }
 }
 }
}

```

```

root++;
int rootItem = root;
tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent
= true, IsExpanded = false, ParentValue = null, Duration = ran.Next(1, 50) });
int parent = root;
for (var d = 0; d < 1; d++)
{
root++;
string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";
int par = parent + 1;
progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
int iD = root;
tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
IsExpanded = false, ParentValue = rootItem, Duration = ran.Next(1, 50) });
int subparent = root;
for (var c = 0; c < 500; c++)
{
root++;
string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";
int subchild = subparent + c + 1;
string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
int childID = root ;
tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(), isParent =
false, IsExpanded = false, ParentValue = subparent, Duration = ran.Next(1, 50) });
}
}
}
}
return tree;
}
}
、

```

### Load parent rows in expanded state with virtualization

Tree Grid provides an option to load the child records in the initial rendering itself for remote data binding by setting the [loadChildOnDemand](#) as true. When the `loadChildOnDemand` is enabled, parent records are rendered in expanded state.

When using virtualization with `loadChildOnDemand`, it helps you to improve the tree grid performance while loading the child records during the initial rendering for remote data binding by setting [enableVirtualization](#) as true and `loadChildOnDemand` as true.

The following code example describes handling of child records at server end.

```
`ts
public ActionResult loadChildOndemand()
{
 TreeData.tree = new List<TreeData>();
 return View();
}

public ActionResult UrlDatasource(DataManagerRequest dm)
{
 List<TreeData> data = new List<TreeData>();
 data = TreeData.GetTree();
 DataOperations operation = new DataOperations();
 IEnumerable<TreeData> DataSource = data;
 if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the ExpandStateMapping
 property whether is true or false
 {
 var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
 val.IsExpanded = false;
 }
 else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction")
 {
 var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
 val.IsExpanded = true;
 }
 if (!dm.Where != null && dm.Where.Count > 1)
 {
 data = data.Where(p => p.ParentValue == null).ToList();
 }
}
```

```
DataSource = data;
if (dm.Search != null && dm.Search.Count > 0) // Searching
{
 DataSource = operation.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting
{
 DataSource = operation.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 1) //filtering
{
 DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");
}
data = new List<TreeData>();
foreach (var rec in DataSource)
{
 data.Add(rec as TreeData);
}
var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.ParentValue)
 .Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
foreach (var Record in data.ToList())
{
 if (GroupData.ContainsKey(Record.TaskID.ToString()))
 {
 var ChildGroup = GroupData[Record.TaskID.ToString()];
 if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting the child records
 {
 IEnumerable ChildSort = ChildGroup;
 ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
 ChildGroup = new List<TreeData>();
 foreach (var rec in ChildSort)
 {
 ChildGroup.Add(rec as TreeData);
 }
 }
 }
}
```



```
}
}
if (dm.Search != null && dm.Search.Count > 0) // Searching the child records
{
 IEnumerable ChildSearch = ChildGroup;
 ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
 ChildGroup = new List<TreeData>();
 foreach (var rec in ChildSearch)
 {
 ChildGroup.Add(rec as TreeData);
 }
}
if (ChildGroup?.Count > 0)
 AppendChildren(dm, ChildGroup, Record, GroupData, data);
}

DataSource = data;

if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the skip index based on
collapsed parent
{
 string IdMapping = "TaskID";
 List<WhereFilter> CollapseFilter = new List<WhereFilter>();
 CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
 var CollapsedParentRecord = operation.PerformFiltering(DataSource, CollapseFilter, "and");
 var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
 dm.Skip = index;
}

else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the skip index based on
expanded parent
{
 string IdMapping = "TaskID";
 List<WhereFilter> ExpandFilter = new List<WhereFilter>();
```

```

ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
var ExpandedParentRecord = operation.PerformFiltering(DataSource, ExpandFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
dm.Skip = index;
}
int count = data.Count;
DataSource = data;
if (dm.Skip != 0)
{
DataSource = operation.PerformSkip(DataSource, dm.Skip); //Paging
}
if (dm.Take != 0)
{
DataSource = operation.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}

private void AppendChildren(DataManagerRequest dm, List<TreeData> ChildRecords, TreeData
ParentValue, Dictionary<string, List<TreeData>> GroupData, List<TreeData> data) // Getting child
records for the respective parent
{
string TaskId = ParentValue.TaskID.ToString();
var index = data.IndexOf(ParentValue);
DataOperations operation = new DataOperations();
foreach (var Child in ChildRecords)
{
if (ParentValue.IsExpanded)
{
string ParentId = Child.ParentValue.ToString();
if (TaskId == ParentId)
{
((IList)data).Insert(++index, Child);
if (GroupData.ContainsKey(Child.TaskID.ToString()))

```

```
{
var DeepChildRecords = GroupData[Child.TaskID.ToString()];
if (DeepChildRecords?.Count > 0)
{
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
{
IEnumerable ChildSort = DeepChildRecords;
ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSort)
{
DeepChildRecords.Add(rec as TreeData);
}
}
if (dm.Search != null && dm.Search.Count > 0) // searching the child records
{
IEnumerable ChildSearch = DeepChildRecords;
ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSearch)
{
DeepChildRecords.Add(rec as TreeData);
}
}
AppendChildren(dm, DeepChildRecords, Child, GroupData, data);
if (Child.IsExpanded)
{
index += DeepChildRecords.Count;
}
}
}
}
```

```
}
}

public ActionResult Update(CRUDModel<TreeData> value)
{
 List<TreeData> data = new List<TreeData>();
 data = TreeData.GetTree();
 var val = data.Where(ds => ds.TaskID == value.Value.TaskID).FirstOrDefault();
 val.TaskName = value.Value.TaskName;
 val.Duration = value.Value.Duration;
 return Json(val);
}

public ActionResult Insert(CRUDModel<TreeData> value)
{
 var c = 0;
 for (; c < TreeData.GetTree().Count; c++)
 {
 if (TreeData.GetTree()[c].TaskID == value.RelationalKey)
 {
 if (TreeData.GetTree()[c].isParent == null)
 {
 TreeData.GetTree()[c].isParent = true;
 }
 break;
 }
 }
 c += FindChildRecords(value.RelationalKey);
 TreeData.GetTree().Insert(c + 1, value.Value);
 return Json(value.Value);
}

public int FindChildRecords(int? id)
{
 var count = 0;
 for (var i = 0; i < TreeData.GetTree().Count; i++)
```

```

{
if (TreeData.GetTree()[i].ParentValue == id)
{
count++;
count += FindChildRecords(TreeData.GetTree()[i].TaskID);
}
}
return count;
}

public object Delete(CRUDModel<TreeData> value)
{
if (value.deleted != null)
{
for (var i = 0; i < value.deleted.Count; i++)
{
TreeData.GetTree().Remove(TreeData.GetTree().Where(ds => ds.TaskID ==
value.deleted[i].TaskID).FirstOrDefault());
}
}
else
{
TreeData.GetTree().Remove(TreeData.GetTree().Where(or => or.TaskID ==
int.Parse(value.Key.ToString())).FirstOrDefault());
}
return Json(value);
}

public class CRUDModel<T> where T : class
{
public TreeData Value;
public int Key { get; set; }
public int RelationalKey { get; set; }
public List<T> added { get; set; }
public List<T> changed { get; set; }
public List<T> deleted { get; set; }
}

```

```

}

public class TreeData
{
 public static List<TreeData> tree = new List<TreeData>();
 [System.ComponentModel.DataAnnotations.Key]
 public int TaskID { get; set; }
 public string TaskName { get; set; }
 public int Duration { get; set; }
 public int? ParentValue { get; set; }
 public bool? isParent { get; set; }
 public bool IsExpanded { get; set; }
 public TreeData() { }
 public static List<TreeData> GetTree()
 {
 if (tree.Count == 0)
 {
 int root = 0;
 for (var t = 1; t <= 1500; t++)
 {
 Random ran = new Random();
 string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
 string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
 root++;
 int rootItem = root;
 tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent = true, IsExpanded = true, ParentValue = null, Duration = ran.Next(1, 50) });
 int parent = root;
 for (var d = 0; d < 1; d++)
 {
 root++;
 string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";
 int par = parent + 1;
 progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";

```

```

int iD = root;

tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
IsExpanded = true, ParentValue = rootItem, Duration = ran.Next(1, 50) });

int subparent = root;

for (var c = 0; c < 6; c++)
{
 root++;

 string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";

 int subchild = subparent + c + 1;

 string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";

 int childID = root ;

 tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(),
 ParentValue = subparent, Duration = ran.Next(1, 50) });

}

}

}

}

return tree;

}

}

`

```

### Immutable mode in EJ2 JavaScript Treegrid control

The immutable mode optimizes the Tree Grid re-rendering performance by using the object reference and [deep compare](#) concept. When performing the Tree Grid actions, it will only re-render the modified or newly added rows and prevent the re-rendering of the unchanged rows.

To enable this feature, you have to set the [enableImmutableMode](#) property as **true**.

\* This feature uses the primary key value for data comparison. So, you need to provide the [isPrimaryKey](#) column.

### INDEX.TS

```

import { TreeGrid, Page, Edit } from '@syncfusion/ej2-treegrid';
import { sampleData2, dataSource1, sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Edit);
let immutableStart: any;
let normalStart: any;
let immutableInit = true;
let init = true;
if(sampleData2.length == 0){
 dataSource1();
}

```

```

let immutableGrid: TreeGrid = new TreeGrid(
{
 dataSource: sampleData2,
 childMapping: 'subtasks',
 height: 350,
 allowPaging: true,
 pageSettings: {pageSize: 50},
 treeColumnIndex: 1,
 enableImmutableMode: true,
 selectionSettings: { type: "Multiple" },
 editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Cell' },
 beforeDataBound: () => {
 immutableStart = new Date().getTime();
 },
 dataBound: ()=> {
 let val = immutableInit ? '' : new Date().getTime() -
immutableStart;
 document.getElementById('immutableDelete').innerHTML =
'Immutable rendering Time: ' + "" + val + "" + 'ms';
 immutableStart = 0; immutableInit = false;
 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey:
true, width: 70, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'endDate', headerText: 'End Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80,
textAlign: 'Right' },
 { field: 'priority', headerText: 'Priority', width: 90 }
]
});
immutableGrid.appendTo('#immutable');
document.getElementById('delete').addEventListener('click', function(e) {
 normalGrid.selectRows([0, 2, 4]);
 immutableGrid.selectRows([0, 2, 4]);
 normalGrid.deleteRecord();
 immutableGrid.deleteRecord();
});
document.getElementById('addtop').addEventListener('click', function(e) {
 normalGrid.addRecord(sampleData[0], 0, "Top");
 immutableGrid.addRecord(sampleData[0], 0, "Top");
});
document.getElementById('addbottom').addEventListener('click', function(e) {
 normalGrid.addRecord(sampleData[0], 0, "Bottom");
 immutableGrid.addRecord(sampleData[0], 0, "Bottom");
});
document.getElementById('reverse').addEventListener('click', function(e) {
 let aData = (immutableGrid.dataSource as any).reverse();
 normalGrid.setProperties({ dataSource: aData });
 immutableGrid.setProperties({ dataSource: aData });
});

```



```

});
document.getElementById('paging').addEventListener('click', function(e) {
 let page = normalGrid.pageSettings.currentPage + 1;
 normalGrid.goToPage(page);
 immutableGrid.goToPage(page);
});
let normalGrid: TreeGrid = new TreeGrid({
 dataSource: sampleData2,
 childMapping: 'subtasks',
 height: 350,
 allowPaging: true,
 selectionSettings: { type: "Multiple" },
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell' },
 pageSettings: { pageSize: 50 },
 treeColumnIndex: 1,
 beforeDataBound: () => {
 normalStart = new Date().getTime();
 },
 dataBound: ()=> {
 let val = init ? '' : new Date().getTime() - normalStart;
 document.getElementById('normalDelete').innerHTML = 'Normal
rendering Time: ' + "" + val + "" + 'ms';
 normalStart = 0; init = false;
 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey:
true, width: 70, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'endDate', headerText: 'End Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80,
textAlign: 'Right' },
 { field: 'priority', headerText: 'Priority', width: 90 }
]
});
normalGrid.appendTo('#normal');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

```

```

<style>
 .immutablegrid,
 .normalgrid {
 pointer-events: none;
 }
</style><script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <table>
 <tbody>
 <tr>
 <td>

 </td>
 </tr>
 <tr>
 <td>

 </td>
 </tr>
 <tr>
 <td>
 <div>
 <button id="addtop" class="e-control e-btn e-lib
e-info">Add row at top</button>
 <button id="addbottom" class="e-control e-btn e-
lib e-info">Add row at bottom</button>
 <button id="delete" class="e-control e-btn e-lib
e-info">Delete 5 rows</button>
 <button id="reverse" class="e-control e-btn e-
lib e-info">Sort Task ID</button>
 <button id="paging" class="e-control e-btn e-lib
e-info">Paging</button>
 </div>
 </td>
 </tr>
 <tr>
 <td>
 Immutable Tree Grid
 <div id="immutable" class="immutablegrid"></div>
 </td>
 <td>
 Normal Tree Grid
 <div id="normal" class="normalgrid"></div>
 </td>
 </tr>
 </tbody>
 </table>
 </div>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Limitations

The following features are not supported in the immutable mode:

- Frozen rows and columns
- Row Template
- Detail Template
- Column reorder
- Virtualization

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

## Columns

Columns in EJ2 JavaScript Treegrid control

The column definitions are used as the [dataSource](#) schema in the TreeGrid. This plays a vital role in rendering column values in the required format.

The treegrid operations such as sorting, filtering and searching etc. are performed based on column definitions. The [field](#) property of the [columns](#)

is necessary to map the data source values in TreeGrid columns.

1. If the column [field](#) is not specified in the dataSource, the column values will be empty.
2. If the [field](#) name contains “dot” operator, it is considered as complex binding.

[treeColumnIndex](#) property denotes the column that is used to expand and collapse child rows.

### Format

To format cell values based on specific culture, use the [columns.format](#) property. The TreeGrid uses [Internalization](#) library to format [number](#) and [date](#)

values.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { formatData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: formatData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'orderId', headerText: 'Order ID', textAlign:
'Right', width: 90 },

```

```

 { field: 'orderName', headerText: 'Order Name', textAlign:
'Left', width: 180 },
 { field: 'price', headerText: 'Price', textAlign: 'Right',
width: 90, format: 'c2', type: 'number' },
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>

```

```

 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the [number](#) and [date](#) values are formatted in **en-US** locale.

#### Number formatting

The number or integer values can be formatted using the below format strings.

Format | Description | Remarks

{ type:'date', format:'dd/MM/yyyy' } | 04/07/1996

{ type:'date', format:'dd.MM.yyyy' } | 04.07.1996

{ type:'date', skeleton:'short' } | 7/4/96

{ type:'dateTime', format:'dd/MM/yyyy hh:mm a' } | 04/07/1996 12:00 AM

{ type:'dateTime', format:'MM/dd/yyyy hh:mm:ss a' } | 07/04/1996 12:00:00 AM

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { formatData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: formatData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'orderID', headerText: 'Order ID', textAlign:
'Right', width: 90 },
 { field: 'orderName', headerText: 'Order Name', textAlign:
'Left', width: 220 },
 {
 field: 'orderDate', headerText: 'Order Date', textAlign:
'Right', width: 160,
 format: { format: 'dd/MM/yyyy', type: 'date' }
 },
 { field: 'price', headerText: 'Price', textAlign: 'Right',
width: 90, format: 'c2', type: 'number' },
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
<meta charset="utf-8">

```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Lock columns

You can lock columns by using [column.lockColumn](#) property. The locked columns will be moved to the first position. Also you can't reorder its position.

In the below example, duration column is locked and its reordering functionality is disabled.

### INDEX.TS

```
import { TreeGrid, Reorder } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Reorder);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowReordering: true,
 allowSelection: false,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
 textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 lockColumn: true, customAttributes: {class: 'customcss'}, textAlign: 'Right'
 }
]
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Column type

Column type can be specified using the [columns.type](#) property. It specifies the type of data the column binds.

If the [format](#) is defined for a column, the column uses [type](#) to select the appropriate format option ([number](#) or [date](#)).

TreeGrid column supports the following types:

- string
- number
- boolean
- date
- datetime

If the [type](#) is not defined, it will be determined from the first record of the [dataSource](#).



*Checkbox column*

To render checkboxes in existing column, you need to set `[columns.showCheckbox]` property as `true`.

It is also possible to select the rows hierarchically using checkboxes in TreeGrid by enabling `[autoCheckHierarchy]` property. When we check on any parent record checkbox then the child record checkboxes will get checked.

**INDEX.TS**

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { data, sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 autoCheckHierarchy: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', showCheckbox:
 true, width: 180 },
 { field: 'approved', headerText: 'Approved', width: 90 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Controlling Tree Grid actions

You can enable or disable treegrid action for a particular column by setting the [allowFiltering](#), and [allowSorting](#) properties.

### INDEX.TS

```

import { TreeGrid, Filter, Sort } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter, Sort);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 270,
 treeColumnIndex: 1,
 allowSorting: true,
 allowFiltering: true,
 columns: [
 {
 field: 'taskID', headerText: 'Task ID', allowSorting:
false, width: 90, textAlign: 'Right'
 },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
]
});

```

```

 field: 'startDate', headerText: 'Start Date',
allowFiltering: false, width: 90, textAlign: 'Right', type: 'date', format:
'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
 <div id="TreeGrid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Show/hide columns by external button

You can show or hide treegrid columns dynamically using external buttons by invoking the [showColumns](#) or [hideColumns](#) method.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 270,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
let show: Button = new Button({ cssClass: 'e-flat' }, '#show');
let hide: Button = new Button({ cssClass: 'e-flat' }, '#hide');
document.getElementById('show').onclick = () => {
 treeGridObj.showColumns(['Task ID', 'Duration']); //show by HeaderText
};
document.getElementById('hide').onclick = () => {
 treeGridObj.hideColumns(['Task ID', 'Duration']); //hide by HeaderText
};

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">

```

```

 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="show">Show</button>
 <button id="hide">Hide</button>
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### ValueAccessor

The [valueAccessor](#) is used to access/manipulate the value of display data. You can achieve custom value formatting by using the [valueAccessor](#).

### INDEX.TS

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { formatData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: formatData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'orderId', headerText: 'Order ID', textAlign: 'Right',
width: 100 },
 {
 field: 'orderName', headerText: 'Order Name', textAlign:
'Left', valueAccessor: orderFormatter, width: 215
 },
 {
 field: 'orderDate', headerText: 'Order Date', textAlign:
'Right', width: 110,
 format: { format: 'dd/MM/yyyy', type: 'date' }
 },
 {
 field: 'price', headerText: 'Price', textAlign: 'Right',
width: 100, valueAccessor: currencyFormatter, type: 'number'
 },
],
 height: 315
});
treeGridObj.appendTo('#TreeGrid');
function currencyFormatter(field: string, data: Object, column: Object):
string {
 return '€' + data['price'];
}
function orderFormatter(field: string, data: Object, column: Object): string
{
 return data[field] + '-' + data['Category'];
}
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Display array type columns

You can bind an array of objects in a column by using the [valueAccessor](#) property.

In this example, the name field has an array of two objects, FirstName and LastName. These two objects are joined and bound to a column using the

[valueAccessor](#).

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { stringData } from './datasource.ts';

```

```

let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: stringData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right',
width: 110 },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'name', headerText: 'Assignee', textAlign: 'Left',
valueAccessor: orderFormatter, width: 150
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
],
 height: 315
});
treeGridObj.appendTo('#TreeGrid');
function orderFormatter(field: string, data: Object, column: Object): string
{
 return data[field].map(function (s: {lastName: string, firstName:
string}): string {
 return s.lastName || s.firstName }).join(' ');
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Expression column

You can achieve the expression column by using the [valueAccessor](#) property.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { formatData } from './datasource.ts';
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: formatData,
 height: 315,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 columns: [
 { field: 'orderID', headerText: 'Order ID', textAlign: 'Right',
width: 110 },
 { field: 'orderName', headerText: 'Order Name', textAlign:
'Left', width: 210 },
 { field: 'units', headerText: 'Units', textAlign: 'Right',
width: 120 },
 { field: 'unitPrice', headerText: 'Unit Price', textAlign:
'Right', width: 120, format: 'c2', type: 'number' },
 { field: 'price', headerText: 'Total Price', valueAccessor:
totalPrice, textAlign: 'Right', width: 120, format: 'c2', type: 'number' },
]
 });
treegrid.appendTo('#TreeGrid');

```

```
function totalPrice(field: string, data: { units: number, Fat: number,
unitPrice: number }, column: Object): number {
 return data.units * data.unitPrice;
};
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
 ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### How to render boolean values as checkbox

To render boolean values as checkbox in columns, you need to set [displayAsCheckBox](#) property as `true`.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { data, sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'approved', headerText: 'Approved',
 displayAsCheckBox: true, width: 90 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

## Headers in EJ2 JavaScript Treegrid control

### Header text

By default, column header title is displayed from column [field](#) value. To override the default header title, you have to define the [headerText](#) value.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },

```

```

 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* If both the [field](#) and [headerText](#) are not defined in the column, the column renders with “empty” header text.

#### Header template

You can customize the header element by using the [headerTemplate](#) property.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { headerData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: headerData,
 childMapping: 'subtasks',
 height: 315,
 columns: [
 { field: 'taskName', headerTemplate: '#projectName', width: 220 },
 { field: 'startDate', headerTemplate: '#dateTemplate', format:
'yMd', textAlign: 'Right' },
 { field: 'duration', headerTemplate: '#durationTemplate', textAlign:
'Right' },
 { field: 'progress', headerTemplate: '#progressTemplate', textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
<script id="template" type="text/x-template">
 <div class="template_checkbox">
 ${if(approved)}
 <input type="checkbox" checked> ${else}
 <input type="checkbox"> ${/if}
 </div>
 </script>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script type="text/x-template" id="projectName">
 <div>
 <div>
 Task Name
 </div>
 </div>
 </script>
 <script type="text/x-template" id="dateTemplate">
 <div>
 <div>
 Start Date
 </div>
 </div>
 </script>

```

```

</script>
<script type="text/x-template" id="durationTemplate">
 <div>
 <div>
 Duration
 </div>
 </div>
</script>
<script type="text/x-template" id="progressTemplate">
 <div>
 <div>
 Progress
 </div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Column template in EJ2 JavaScript Treegrid control

The column [template](#) has options to display custom element instead of a field value in the column.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { textdata, getSparkData } from './datasource.ts';
import { Sparkline } from '@syncfusion/ej2-charts';
import { RowDataBoundEventArgs, getObject } from '@syncfusion/ej2-grids';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: textdata,
 childMapping: 'Children',
 treeColumnIndex: 0,
 rowDataBound: (args: RowDataBoundEventArgs) : void => {
 let data: string = getObject('EmployeeID', args.data);
 let spkwl: HTMLElement = args.row.querySelector('#spkwl' + data);
 let winloss: Sparkline = new Sparkline({
 height: '50px',
 width: '150px',
 type: 'WinLoss',
 valueType: 'Numeric',
 fill: '#3C78EF',
 tiePointColor: 'darkgray',
 negativePointColor: '#f7a816',
 dataSource: getSparkData('column', +data)
 });
 winloss.appendTo(spkwl);
 },
 rowHeight: 83,
 columns: [

```



```

 { field: 'EmpID', headerText: 'Employee ID', width: 95 },
 { field: 'Name', headerText: 'Name', width: 110 },
 { field: 'DOB', headerText: 'DOB', width: 90, textAlign: 'Right',
format: 'yMd' },
 {
 headerText: 'Year GR', textAlign: 'Center',
 template: '#template', width: 120
 }
]
 height: 260
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>

```

```

</head>
<body>
 <script type="text/x-template" id="template">
 <div id="spkwl${EmployeeID}"></div>
 </script>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

TreeGrid actions such as editing, filtering and sorting etc. will depend upon the column [field](#). If the [field](#) is not specified in the template column, the treegrid actions cannot be performed.

#### Using condition template

You can render the template elements based on condition.

In the following code, checkbox is rendered based on **Approved** field value.

```

<script id="template" type="text/x-template">
<div class="template_checkbox">
 ${if(approved)}
<input type="checkbox" checked> ${else}
<input type="checkbox"> ${/if}
</div>
</script>

```

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowResizing: true,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'taskName', headerText: 'Task Name', width: 180 },

```

```

 {
 headerText: 'Approved', textAlign: 'Center',
 template: '#template', width: 120
 },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <script id="template" type="text/x-template">

```

```

 <div class="template_checkbox">
 ${if(approved)}
 <input type="checkbox" checked> ${else}
 <input type="checkbox"> ${/if}
 </div>
 </script>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Complex data binding in EJ2 JavaScript Treegrid control

You can achieve complex data binding in the treegrid by using the dot(.) operator in the [column.field](#).

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { complexData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: complexData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'assignee.firstName', headerText: 'Assignee',
 width: 90},
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Auto fit columns in EJ2 JavaScript Treegrid control

The [autoFitColumns](#) method resizes the column to fit the widest cell's content without wrapping. You can autofit a specific column at initial rendering by invoking the [autoFitColumns](#) method in [dataBound](#) event.

### INDEX.TS

```

import { TreeGrid, Resize } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';

```

```

TreeGrid.Inject(Resize);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowResizing: true,
 dataBound: () => treeGridObj.autoFitColumns(['taskName']),
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'taskName', headerText: 'Task Name', width: 60 },
 {
 field: 'startDate', headerText: 'Start Date', format: 'yMd',
width: 120, textAlign: 'Right'
 },
 { field: 'duration', headerText: 'Duration', width: 120, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 120, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can autofit all the columns by invoking the [autoFitColumns](#) method without column names.

#### Column reorder in EJ2 JavaScript Treegrid control

Reordering can be done by drag and drop of a particular column header from one index to another index within the treegrid. To enable reordering, set the [allowReordering](#) to true.

To use reordering, inject the [Reorder](#) module in the treegrid.

#### INDEX.TS

```

import { TreeGrid, Reorder } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Reorder);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowReordering: true,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

You can disable reordering a particular column by setting the [columns.allowReordering](#) to false.

### Reorder multiple columns

Multiple columns can be reordered at a time by using the [reorderColumns](#) method.

### INDEX.TS

```
import { TreeGrid, Reorder } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
TreeGrid.Inject(Reorder);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowReordering: true,
 height: 285,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
let reorderMultipleColsBtn: Button = new Button();
reorderMultipleColsBtn.appendTo('#reorderMultipleCols');
document.getElementById('reorderMultipleCols').addEventListener('click', ()
=> {
 treeGridObj.reorderColumns(['taskID', 'duration'], 'progress');
});
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="reorderMultipleCols">Reorder Task ID and Duration to
Last</button>
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Column resizing in EJ2 JavaScript Treegrid control

Column width can be resized by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. Each column can be auto resized by double-clicking the right edge of the column header to fit the width of that column based on the widest cell content. To enable column resize, set the [allowResizing](#) property to true.

To use the column resize, inject **Resize** module in the treegrid.

### INDEX.TS

```

import { TreeGrid, Resize } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Resize);

```

```

let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowResizing: true,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* You can disable resizing for a particular column by setting the [columns.allowResizing](#) to false.

\* In RTL mode, you can click and drag the left edge of the header cell to resize the column.

#### *Min and max width*

Column resize can be restricted between minimum and maximum width by defining the [columns->minWidth](#) and [columns->maxWidth](#).

In the following sample, minimum and maximum width are defined for **Duration**, and **Task Name** columns.

#### **INDEX.TS**

```

import { TreeGrid, Resize } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Resize);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowResizing: true,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'taskName', headerText: 'Task Name', minWidth: 170,
maxWidth: 250, width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, minWidth:
50, maxWidth: 150, textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### **INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

*Resize stacked column*

Stacked columns can be resized by clicking and dragging the right edge of the stacked column header. While dragging, the width of the respective child columns will be resized at the same time. You can disable resize for particular stacked column by setting `allowResizing` as `false` to its columns.

**INDEX.TS**

```
import { TreeGrid, Resize } from '@syncfusion/ej2-treegrid';
import { stackedData } from './datasource.ts';
TreeGrid.Inject(Resize);
let treeGridObj: TreeGrid = new TreeGrid(
 {
 dataSource: stackedData,
 childMapping: 'subtasks',
 allowResizing: true,
 treeColumnIndex: 1,
 height: 260
 columns: [
 {
 headerText: 'Order Details', textAlign: 'Center', columns: [
 { field: 'orderID', headerText: 'Order ID', textAlign:
'Right', width: 90 },
 { field: 'orderName', headerText: 'Order Name',
textAlign: 'Left', width: 170 },
]
 },
 {
 headerText: 'Shipment Details', textAlign: 'Center',
columns: [
 { field: 'shipMentCategory', headerText: 'Shipment
Category', textAlign: 'Left', width: 90 },
 { field: 'shippedDate', headerText: 'Shipped Date',
textAlign: 'Right', width: 90, format: 'yMd' }
]
 }
]
 });
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

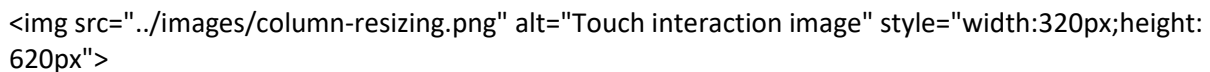
 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Touch interaction

When the right edge of the header cell is tapped, a floating handler will be visible over the right border of the column. To resize the column, tap and drag the floating handler as needed.

The following screenshot represents the column resizing in touch device.

The image shows a touch interaction for column resizing. It displays a grid with a floating handler on the right edge of a header cell, which is used to tap and drag to resize the column.

### Column chooser in EJ2 JavaScript Treegrid control

The column chooser has options to show or hide columns dynamically. It can be enabled by defining the [showColumnChooser](#) as true.

To use the column chooser, inject the **Column Chooser** module in the treegrid.

**INDEX.TS**

```
import { TreeGrid, Selection, Toolbar, ColumnChooser } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Selection, Toolbar, ColumnChooser);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 showColumnChooser: true,
 treeColumnIndex: 1,
 toolbar: ['ColumnChooser'],
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },
 { field: 'taskName', headerText: 'Task Name', width: 240,
showInColumnChooser: false },
 { field: 'startDate', headerText: 'Start Date', width: 110, format:
'yMd' },
 { field: 'endDate', headerText: 'End Date', width: 110, textAlign:
'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 100, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 100, textAlign:
'Right' },
 { field: 'priority', headerText: 'Priority', width: 90 }
]
 height: 315
});
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can hide the column names in column chooser by defining the [columns.showInColumnChooser](#) as false.

#### *Open column chooser by external button*

The Column chooser can be displayed on a page through external button by invoking the [openColumnChooser](#) method with X and Y axis positions.

#### **INDEX.TS**

```

import { TreeGrid, Selection, ColumnChooser } from '@syncfusion/ej2-
treegrid';
import { sampleData } from './datasource.ts';
import { Button } from '@syncfusion/ej2-buttons';
TreeGrid.Inject(Selection, ColumnChooser);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 showColumnChooser: true,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign: 'Right', width:
90 },

```

```

 { field: 'taskName', headerText: 'Task Name', width: 240,
showInColumnChooser: false },
 { field: 'startDate', headerText: 'Start Date', width: 110, format:
'yMd' },
 { field: 'endDate', headerText: 'End Date', width: 110, textAlign:
'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 100, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 100, textAlign:
'Right' },
 { field: 'priority', headerText: 'Priority', width: 90 }
]
 height: 315
});
treeGridObj.appendTo('#TreeGrid');
let show: Button = new Button({ cssClass: 'e-flat' }, '#show');
document.getElementById('show').onclick = () => {
 treeGridObj.columnChooserModule.openColumnChooser(200, 50); // give X
and Y axis
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="show">Open Column Chooser</button>
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Column menu in EJ2 JavaScript Treegrid control

The column menu has options to integrate features like sorting, filtering, and autofit. It will show a menu with the integrated feature when users click on multiple icon of the column. To enable column menu, you need to define the [showColumnMenu](#) property as true.

By default, column menu is enabled for all columns and you can disable column menu for a particular column by defining the [showColumnMenu](#) as false in [columns](#) property.

To use the column menu, inject the `ColumnMenu` module in the treegrid.

The default items are displayed in following table.

Item	Description
----- -----	
<code>SortAscending</code>	Sort the current column in ascending order.
<code>SortDescending</code>	Sort the current column in descending order.
<code>AutoFit</code>	Auto fit the current column.
<code>AutoFitAll</code>	Auto fit all columns.
<code>Filter</code>	Show the filter option as given in <code>filterSettings.type</code>

### INDEX.TS

```

import {TreeGrid, Sort, Page, Filter, Resize, ColumnMenu} from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Filter, Sort, Resize, ColumnMenu);
let treegrid: TreeGrid = new TreeGrid(
{

```

```

 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 allowFiltering: true,
 allowResizing: true,
 filterSettings: { type: 'Menu' },
 allowSorting: true,
 showColumnMenu: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', showColumnMenu: false, type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
 });
 treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

#### Custom column menu item

The custom column menu items can be added to the column menu by defining the [columnMenuItems](#) as a collection of the [MenuItemModel](#).

The action for custom column menu items can be performed using [columnMenuClick](#) event.

In the below example, clear sorting action was performed through [columnMenuItems](#) by using [clearSorting](#) method of the Tree Grid.

#### INDEX.TS

```

import {TreeGrid, ColumnMenu, Sort} from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(ColumnMenu, Sort);
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 showColumnMenu: true,
 allowSorting: true,
 sortSettings: {
 columns: [{ direction: 'Ascending', field: 'taskID' }],
 },
 columnMenuItems: [{ text: 'Clear Sorting', id: 'clearsorting' }],
 columnMenuClick: function (args) {
 if (args.item.id === 'clearsorting') {
 treegrid.clearSorting();
 }
 },
 },

```

```

 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
 });
 treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Customize menu items for particular columns

It is possible to customize specific items from the column menu for particular [columns](#) using [columnMenuOpen](#) event. `columnMenuOpen` event can be utilized to determine which items to customize in column menu.

The following example shows how to hide the built-in **Filter** menu item when the column menu is opened for the **taskName** column, while allowing it to remain visible for all other columns.

### INDEX.TS

```

import {TreeGrid, ColumnMenu, Sort, Filter} from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
import { ColumnMenuOpenEventArgs, ColumnMenuItemModel } from '@syncfusion/ej2-grids';
TreeGrid.Inject(ColumnMenu, Sort, Filter);
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 showColumnMenu: true,
 allowSorting: true,
 allowFiltering: true,
 filterSettings: { type: 'Menu' },
 treeColumnIndex: 1,
 columnMenuOpen: function (args) {
 for (const item of args.items) {
 if (item.text === 'Filter' && args.column.field ===
'taskName') {
 (item as ColumnMenuItemModel).hide = true;
 } else {
 (item as ColumnMenuItemModel).hide = false;
 }
 }
 },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },

```

```

 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```



```

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Responsive columns in EJ2 JavaScript Treegrid control

You can toggle column visibility based on media queries which are defined

at the [hideAtMedia](#).

The [hideAtMedia](#) accepts valid

[Media Queries](#).

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 1,
 columns: [
 {
 field: 'taskID', headerText: 'Task ID', hideAtMedia:
'(min-width: 700px)', width: 90, textAlign: 'Right'
 }, // column hides when browser screen width less than 700px;
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date',
hideAtMedia: '(max-width: 500px)', width: 90, textAlign: 'Right', type:
'date', format: 'yMd'
 }, // column shows when browser screen width less than or
equal to 500px;
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">

```

```

 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Row

Row in EJ2 JavaScript Treegrid control

The row represents record details fetched from data source.

### Customize rows

You can customize the appearance of a row by using the [rowDataBound](#) event.

The [rowDataBound](#) event triggers for every row. In the event handler, you can get the [RowDataBoundEventArgs](#) that contains details of the row.

### INDEX.TS

```
import { TreeGrid, RowDataBoundEventArgs } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 treeColumnIndex: 1,
 rowDataBound: rowBound,
 enableHover: false,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
function rowBound(args: RowDataBoundEventArgs) {
 if (args.data['duration'] == 0) {
 args.row.style.background= '#336c12';
 } else if (args.data['duration'] < 3) {
 args.row.style.background= '#7b2b1d';
 }
}
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Styling alternate rows

You can change the treegrid's alternative rows' background color by overriding the `.e-altrow` class.

```

.e-treegrid .e-altrow {
background-color: #fafafa;
}

```

Please refer to the following example.

### INDEX.TS

```
import { TreeGrid, RowDataBoundEventArgs } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 treeColumnIndex: 1,
 enableHover: false,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Indent in EJ2 JavaScript Treegrid control

The Indent and Outdent feature will help to change the hierarchy level of rows in tree grid. The indent action moves the selected row as the last child of its previous row, whereas the outdent action moves the selected row as a sibling to its parent row.

To use the indent and outdent feature, inject the **RowDD** module in the Tree Grid. The tree grid toolbar has the built-in items to execute indent and outdent actions. Define this by using the toolbar property.

#### INDEX.TS

```

import { TreeGrid, RowDD, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(RowDD, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Indent', 'Outdent'],
 selectedRowIndex: 2,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'priority', headerText: 'Priority', width: 90 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270

```

```
});
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
```

```

 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### *Indent/Outdent a row programmatically*

You can change the hierarchy level of record programmatically using [indent](#) and [outdent](#) methods.

### **INDEX.TS**

```

import { TreeGrid, RowDD } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(RowDD);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'priority', headerText: 'Priority', width: 90 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
document.getElementById('Indenting').addEventListener('click', Indent);
document.getElementById('Outdenting').addEventListener('click', Outdent);
function Indent() {
 treeGridObj.indent(treeGridObj.getCurrentViewRecords()[2]);
}
function Outdent() {
 treeGridObj.outdent(treeGridObj.getCurrentViewRecords()[2]);
}

```

### **INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
 splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
 </head>
 <body>

 <button id="Indenting">Indent</button>
 <button id="Outdenting">Outdent</button>
 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
 var ele = document.getElementById('container');
 if(ele) {
 ele.style.visibility = "visible";
 }
 </script>
 <script src="index.js" type="text/javascript"></script>
 </body></html>

```

### Row height in EJ2 JavaScript Treegrid control

You can customize the row height of treegrid rows through the [rowHeight](#) property. The `rowHeight` property is used to change the row height of entire treegrid rows.

In the below example, the `rowHeight` is set as '60px'.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',

```

```

height: 275,
rowHeight: 60,
treeColumnIndex: 1,
columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Customize row height for particular row

Grid row height for particular row can be customized using the [rowDataBound](#) event by setting the **rowHeight** in arguments for each row based on the requirement.

In the below example, the row height for the row with Task ID as '3' is set as '90px' using the **rowDataBound** event.

### INDEX.TS

```

import { TreeGrid, RowDataBoundEventArgs } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 rowDataBound: rowBound,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
function rowBound(args: RowDataBoundEventArgs) {
 if((args.data as TaskDetails).taskID === 3){
 args.rowHeight = 90;
 }
}
interface TaskDetails {

```

```
taskID ?: number;
}
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Row template in EJ2 JavaScript Treegrid control

The [rowTemplate](#) has an option to customise the look and behavior of the treegrid rows. The [rowTemplate](#) property accepts either the template string or HTML element ID.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { textdata } from './datasource.ts';
let treegrid: TreeGrid = new TreeGrid(
{
 dataSource: textdata,
 childMapping: 'Children',
 treeColumnIndex: 0,
 rowTemplate: '#rowtemplate',
 height: 250,
 width: 'auto',
 rowHeight: 83,
 columns: [
 { field: 'EmpID', headerText: 'Employee ID', width: '150' },
 { field: 'Name', headerText: 'Employee Name', width: '150' },
 { field: 'Address', headerText: 'Employee Details', width:
'350' }
]
});
treegrid.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 TreeGrid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript TreeGrid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <script id="rowtemplate" type="text/x-template">
 <tr>
 <td class="border" style='padding-left:18px;' >
 <div>${EmpID}</div>
 </td>
 <td class="border" style='padding: 10px 0px 0px 20px;'>
 <div style="font-size:14px;">
 ${Name}
 <p style="font-size:9px;">${Designation}</p>
 </div>
 </td>
 <td class="border">
 <div>
 <div style="position:relative;display:inline-
block;">

 </div>
 <div style="display:inline-block;">
 <div style="padding:5px;">${Address}</div>
 <div style="padding:5px;">${Country}</div>
 <div style="padding:5px;font-
size:12px;">${Contact}</div>
 </div>
 </div>
 </td>
 </tr>
 </script>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</script>
var ele = document.getElementById('container');
if(ele) {

```

```

 ele.style.visibility = "visible";
 }
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The [rowTemplate](#) property accepts only the TR element.

#### Row template with formatting

If the [rowTemplate](#) is used, the value cannot be formatted inside the template using the [columns.format](#) property. In that case, a function should be defined globally to format the value and invoke it inside the template.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { textdata } from './datasource.ts';
import { Internationalization } from '@syncfusion/ej2-base';
let instance: Internationalization = new Internationalization();
(window as DateFormat).format = (value: Date) => {
 return instance.formatDate(value, { skeleton: 'yMd', type: 'date' });
};
interface DateFormat extends Window {
 format?: Function;
}
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: textdata,
 childMapping: 'Children',
 treeColumnIndex: 0,
 rowTemplate: '#rowtemplate',
 height: 250,
 width: 'auto',
 rowHeight: 83,
 columns: [
 { field: 'EmpID', headerText: 'Employee ID', width: '150' },
 { field: 'Address', headerText: 'Employee Details', width:
'350' },
 { field: 'DOB', headerText: 'DOB', width: '150' }
]
 });
treegrid.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <script id="rowtemplate" type="text/x-template">
 <tr>
 <td class="border" style='padding-left:18px;' >
 <div>${EmpID}</div>
 </td>
 <td class="border">
 <div>
 <div style="position:relative;display:inline-
block;">

 </div>
 <div style="display:inline-block;">
 <div style="padding:5px;">${Address}</div>
 <div style="padding:5px;">${Country}</div>
 <div style="padding:5px;font-
size:12px;">${Contact}</div>
 </div>
 </div>
 </td>
 <td class="border" style='padding-left: 20px;'>
 <div>${format (data.DOB) }</div>
 </td>
 </tr>
 </script>

```



```

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Limitations

Row template feature is not compatible with all the features which are available in treegrid and it has limited features support. Here we have listed out the features which are not compatible with row template feature.

- Filtering
- Paging
- Sorting
- Scrolling
- Searching
- Rtl
- Context Menu
- State Persistence

### Detail template in EJ2 JavaScript Treegrid control

The detail template provides additional information about a particular row. By expanding the parent row the child rows are expanded along with their detail template. The [detailTemplate](#) property accepts either the template string or HTML element ID.

To use detail template, inject the DetailRow module in the TreeGrid.

### INDEX.TS

```

import { TreeGrid, DetailRow } from '@syncfusion/ej2-treegrid';
import { textdata } from './datasource.ts';
import { Internationalization } from '@syncfusion/ej2-base';
TreeGrid.Inject(DetailRow);
let instance: Internationalization = new Internationalization();
(window as DateFormat).format = (value: Date) => {
 return instance.formatDate(value, { skeleton: 'yMd', type: 'date' });
};
interface DateFormat extends Window {
 format?: Function;
}
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: textdata,
 childMapping: 'Children',
 treeColumnIndex: 0,
 detailTemplate: '#detailtemplate',
 }
);

```

```

 width: 'auto',
 columns: [
 { field: 'Name', headerText: 'First Name', width: '170' },
 { field: 'Designation', headerText: 'Designation', width:
'180' },
 { field: 'EmpID', headerText: 'EmployeeID', width: '110'},
 { field: 'Country', headerText: 'Country' , width: '90'},
]
 });
 treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <script id="detailtemplate" type="text/x-template">
 <div style="position: relative; display: inline-block; float:
left; font-weight: bold; width: 10%;padding:5px 4px 2px 27px; ">

 </div>
 </script>

```

```

 <div style="padding-left: 10px; display: inline-block; width:
66%; text-wrap: normal;font-size:13px;font-family:'Segoe UI';">
 <div class="e-description" style="word-wrap: break-word;">
 ${Name} was born on ${format(data.DOB)}. Now
lives at ${Address}, ${Country}. ${Name} holds a position of
${Designation} in our WA department, (Seattle USA).
 </div>
 <div class="e-description" style="word-wrap: break-
word;margin-top:5px;">
 <b style="margin-right:10px;">Contact:${Contact}
 </div>
 </div>
 </script>
 <div class="loader">
 <div class="container">
 <div id="TreeGrid">
 </div>
 </div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Row drag and drop in EJ2 JavaScript Treegrid control

The TreeGrid rows can be reordered, dropped to another TreeGrid or custom control by enabling the [allowRowDragAndDrop](#) to true.

To use row drag and drop, inject the **RowDD** module in the TreeGrid.

#### *Drag and drop within Tree Grid*

The TreeGrid row drag and drop allows you to drag and drop TreeGrid rows on the same TreeGrid using drag icon. To enable row drag and drop, set the [allowRowDragAndDrop](#) to true. It provides the way to drop the row above, below or child to the target row with respective to the target row position.

### **INDEX.TS**

```

import { TreeGrid, RowDD } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(RowDD);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 treeColumnIndex: 1,
 childMapping: 'subtasks',
 allowRowDragAndDrop: true,
 height: 300,
 selectionSettings: { type: 'Multiple' },
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {

```

```

 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">

```

```

 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* Selection feature must be enabled for row drag and drop.

\* For multiple row selection, the [type](#) property must be set to `multiple`.

\* The [isPrimaryKey](#) property is necessary to perform row drag and drop operation.

#### *Drag and drop to another Tree Grid*

To drag and drop between two TreeGrid, enable the [allowRowDragAndDrop](#) property and specify the target TreeGrid ID in [targetID](#) property of rowDropSettings.

#### **INDEX.TS**

```

import { TreeGrid, RowDD } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(RowDD);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 treeColumnIndex: 1,
 childMapping: 'subtasks',
 allowRowDragAndDrop: true,
 rowDropSettings: { targetID: 'destTree' },
 selectionSettings: { type: 'Multiple' },
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treegrid.appendTo('#TreeGrid');
let destTree: TreeGrid = new TreeGrid({
 dataSource: [],
 treeColumnIndex: 1,
 childMapping: 'subtasks',
 allowRowDragAndDrop: true,
 rowDropSettings: { targetID: 'TreeGrid' },
 selectionSettings: { type: 'Multiple' },
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
]
});

```

```

 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
destTree.appendTo('#destTree');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
 <div>
 <div style="float: left; width:49%" id="TreeGrid"></div>
 <div style="float: right; width:49%" id="destTree"></div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Drag and drop events

The following events are triggered while drag and drop the treegrid rows.

[rowDragStartHelper](#) - Triggers when click the drag icon or treegrid row and this event is used to customize the drag element based on user criteria.<br/>

[rowDragStart](#) -Triggers when starts to drag the treegrid row. <br/>

[rowDrag](#) - Triggers while dragging the treegrid row. <br/>

[rowDrop](#) - Triggers when a drag element is dropped on the target element. <br/>

### Prevent reordering a row as child to another row

You can prevent the default behavior of dropping rows as children to the target by setting the `cancel` property to `true` in [rowDrop](#) event argument. You can also change the drop position after cancelling using [reorderRows](#) method.

In the below example drop action is cancelled and dropped above to target row.

### INDEX.TS

```

import { TreeGrid, RowDD } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(RowDD);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 treeColumnIndex: 1,
 childMapping: 'subtasks',
 allowRowDragAndDrop: true,
 height: 300,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
],
 rowDrop : (args) => {

```

```

 if (args.dropPosition == 'middleSegment') {
 args.cancel = true;
 treegrid.reorderRows([args.fromIndex], args.dropIndex, 'above');
 }
 });
 treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>

```



```

 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Cell

Cell in EJ2 JavaScript Treegrid control

### Customize cell styles

The appearance of cells can be customized by using the [queryCellInfo](#) event.

The [queryCellInfo](#) event triggers for every cell. In that event handler, you can get [QueryCellInfoEventArgs](#) that contains the details of the cell.

### INDEX.TS

```

import { TreeGrid, QueryCellInfoEventArgs } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 300,
 queryCellInfo: customiseCell,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
function customiseCell(args: QueryCellInfoEventArgs) {
 if (args.column.field === 'progress' && +args.cell.innerHTML > 90 &&
+args.cell.innerHTML <= 100) {
 args.cell.setAttribute('style', 'background-
color:#336c12;color:white;');
 } else if (+args.cell.innerHTML > 20 && args.column.field ===
'progress') {
 args.cell.setAttribute('style', 'background-
color:#7b2b1d;color:white;');
 }
}

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>

```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Custom attributes

You can customize the treegrid cells by adding a CSS class to the [customAttribute](#) property of the column.

```
`css
.e-attr {
background: '#d7f0f4';
}
`ts
{
field: 'taskID', headerText: 'Task ID', width: 90, customAttributes: {class: "e-attr"}, textAlign: 'Right'
}
`
```

In the below example, we have customized the cells of **TaskID** and **StartDate** columns.

### INDEX.TS

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 300,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
customAttributes: {class: "e-attr"}, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 160 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
customAttributes: {class: "e-attr"}, textAlign: 'Right', type:
'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
 .e-attr {
 background: #d7f0f4;
 }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Grid lines

The [gridLines](#) have option to display cell border and it can be defined by the [gridLines](#) property.

The available modes of grid lines are:

- | Modes | Actions |
- |-----|-----|
- | Both | Displays both the horizontal and vertical grid lines.|
- | None | No grid lines are displayed.|
- | Horizontal | Displays the horizontal grid lines only.|
- | Vertical | Displays the vertical grid lines only.|
- | Default | Displays grid lines based on the theme.|

### INDEX.TS

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 300,
 gridLines: 'Both',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 160 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, the treegrid renders with **Default** mode.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Content in EJ2 JavaScript Treegrid control

The HTML tags can be displayed in the TreeGrid header and content by enabling the [disableHtmlEncode](#) property.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { htmlData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: htmlData,
 childMapping: 'subtasks',

```

```

 height: 300,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: ' Task ID ',
 disableHtmlEncode: true, width: 140, textAlign: 'Right' },
 { field: 'taskName', headerText: ' Task Name ',
 disableHtmlEncode: false, width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
 'Right' }
]
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Auto wrap in EJ2 JavaScript Treegrid control

The auto wrap allows the cell content of the treegrid to wrap to the next line when it exceeds the boundary of the cell width. The Cell Content wrapping works based on the position of white space between words.

To enable auto wrap, set the [allowTextWrap](#) property to **true**.

You can configure the auto wrap mode by setting the [textWrapSettings.wrapMode](#) property.

There are three types of **wrapMode**. They are:

- **Both:** Both value is set by default. Auto wrap will be enabled for both the content and the header.
- **Header:** Auto wrap will be enabled only for the header.
- **Content:** Auto wrap will be enabled only for the content.

Note: When a column width is not specified, then auto wrap of columns will be adjusted with respect to the treegrid's width.

In the following example, the **textWrapSettings.wrapMode** is set to **Content**.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 300,
 allowTextWrap: true,
 textWrapSettings: { wrapMode: 'Content' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 75 },
 {

```



```

 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">

```

```

 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Clip mode in EJ2 JavaScript Treegrid control

The clip mode provides options to display its overflow cell content and it can be defined by the [columns.clipMode](#) property.

There are three types of [clipMode](#). They are:

- **Clip:** Truncates the cell content when it overflows its area.
- **Ellipsis:** Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip:** Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { complexData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: complexData,
 childMapping: 'subtasks',
 height: 300,
 gridLines: 'Both',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', clipMode: 'Clip',
width: 70 },
 { field: 'assignee.firstName', headerText: 'Assignee', clipMode:
'Ellipsis', width: 30},
 { field: 'priority', headerText: 'Priority', clipMode:
'EllipsisWithTooltip', width: 30 },
 { field: 'duration', headerText: 'Duration', width: 30, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, [columns.clipMode](#) value is `Ellipsis`.

## Editing

### Edit in EJ2 JavaScript Treegrid control

The TreeGrid component has options to dynamically insert, delete and update records.

Editing feature is enabled by using [editSettings](#) property and it requires a primary key column for CRUD operations.

To define the primary key, set [columns.isPrimaryKey](#) to `true` in particular column.

To use CRUD, inject the [Edit](#) module in treegrid.

#### INDEX.TS

```
import { TreeGrid, Edit } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'priority', headerText: 'Priority', width: 90 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```

```
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

\* You can disable editing for a particular column, by specifying [columns.allowEditing](#) to `false`.

#### *Toolbar with edit option*

The treegrid toolbar has the built-in items to execute Editing actions.

You can define this by using the [toolbar](#) property.

#### **INDEX.TS**

```
import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Row' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'priority', headerText: 'Priority', width: 90 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
```

#### **INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
```

```

<link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

#### Adding row position

The TreeGrid control provides the support to add the new row in the top, bottom, above selected row, below selected row and child position of tree grid content using [editSettings.newRowPosition](#) property. By default, a new row will be added at the top of the treegrid.

The following examples shows how to set new row position as **Child** in tree grid.

#### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, newRowPosition: 'Child', mode: 'Cell' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', editType: 'datepickeredit', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],

```

```

 height: 270
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

### Confirmation messages

#### Delete confirmation

The delete confirm dialog can be shown when deleting a record by defining the [showDeleteConfirmDialog](#) as `true`

## INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],

```

```

editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, showDeleteConfirmDialog: true, mode: 'Cell' },
treeColumnIndex: 1,
columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right'},
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', editType: 'datepickeredit', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Reight' }
],
height: 270
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

The `showDeleteConfirmDialog` supports all type of edit modes.



*Default column values on add new*

The treegrid provides an option to set the default value for the columns when adding a new record in it.

To set a default value for the particular column by defining the [columns.defaultValue](#).

**INDEX.TS**

```
import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, showDeleteConfirmDialog: true, mode: 'Cell' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right'},
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', editType: 'datepickeredit', type: 'date', format: 'yMd'
 },
 { field: 'priority', headerText: 'Priority', width: 80,
 defaultValue: 'Normal' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
```

```

 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Disable editing for particular column

You can disable editing for particular columns by using the [columns.allowEditing](#).

In the following demo, editing is disabled for the **Start Date** column.

### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
allowEditing: false, textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

#### *Troubleshoot: Editing works only for first row*

The Editing functionalities can be performed based upon the primary key value of the selected row.

If `primaryKey` is not defined in the treegrid, then edit or delete action take places the first row.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to knows how to present and manipulate data.

Edit types in EJ2 JavaScript Treegrid control

#### *Cell edit type and its params*

The `columns.editType` is used to customize the edit type of the particular column.

You can set the `columns.editType` based on data type of the column.

- `numericedit` - [NumericTextBox](#) component for integers, double, and decimal data types.
- `defaultedit` - [TextBox](#) component for string data type.
- `dropdownedit` - [DropDownList](#) component for list data type.
- `booleanedit` - [CheckBox](#) component for boolean data type.
- `datepickeredit` - [DatePicker](#) component for date data type.
- `datetimepickeredit` - [DateTimePicker](#) component for date time data type.

Also, you can customize model of the `columns.editType` component through the `columns.edit.params`.

The following table describes cell edit type component and their corresponding edit params of the column.

Component | Example

[NumericTextBox](#) | params: { decimals: 2, value: 5 }

[TextBox](#) | -

[DropDownList](#) | params: { value: 'Germany' }

[Checkbox](#) | params: { checked: true }

[DatePicker](#) | params: { format: 'dd.MM.yyyy' }

[DateTimePicker](#) | params: { value: new Date() }

## INDEX.TS

```
import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Row' },
 treeColumnIndex: 1,
 columns: [
 {
 field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
textAlign: 'Right', width: 100
 },
 {
 field: 'taskName', headerText: 'Task Name', editType:
'stringedit', width: 170
 },
 {
 field: 'startDate', headerText: 'Start Date', textAlign:
'Right', width: 180,
 editType: 'datetimepickeredit', edit: { params: { format: 'M/d/y
hh:mm a' } },
 format: { format: 'M/d/y hh:mm a', type: 'dateTime' }
 },
 {
 field: 'approved', headerText: 'Approved', width: 110, editType:
'booleanedit',
 type: 'boolean', displayAsCheckBox: true
 },
 {
 field: 'progress', headerText: 'Progress', textAlign: 'Right',
width: 120,
 editType: 'numericedit', edit: { params: { format: 'n' } }
 },
 { field: 'priority', headerText: 'Priority', width: 110, editType:
'dropdownedit' }
]
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Tree Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Tree Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<!-- Syncfusion Essential JS 2 Styles -->
<link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If edit type is not defined in the column, then it will be considered as the **stringedit** type (Textbox component).

#### Cell edit template

The cell edit template is used to add a custom component for a particular column by invoking the following functions:

- **create** - It is used to create the element at the time of initialization.
- **write** - It is used to create the custom component or assign default value at the time of editing.
- **read** - It is used to read the value from the component at the time of save.
- **destroy** - It is used to destroy the component.

#### INDEX.TS

```

import { TreeGrid, Edit, Toolbar, Column } from '@syncfusion/ej2-treegrid';
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let elem: HTMLElement;
let autoCompleteObj: AutoComplete;
let treeGridObj: TreeGrid = new TreeGrid(

```

```

{
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 height: 400,
 editSettings: {
 allowAdding: true,
 allowEditing: true,
 allowDeleting: true,
 mode: 'Cell',
 newRowPosition: 'Below'
 },
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 columns: [
 {
 field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
 textAlign: 'Right', width: 90
 },
 { field: 'taskName', headerText: 'Task Name', editType:
'stringedit', edit: {
 create: () => {
 elem = document.createElement('input');
 return elem;
 },
 read: () => {
 return autoCompleteObj.value;
 },
 destroy: () => {
 autoCompleteObj.destroy();
 },
 write: (args: { rowData: Object, column: Column }) => {
 autoCompleteObj = new AutoComplete({
 dataSource: <{key: string, value:
any}[]>treeGridObj.grid.dataSource,
 fields: { value: 'taskName' },
 value: args.rowData[args.column.field]
 });
 autoCompleteObj.appendTo(elem);
 }
 },
 width: 180 },
 { field: 'startDate', headerText: 'Start Date', textAlign:
'Right', width: 130, editType: 'datepickeredit', type: 'date', format: 'yMd'
 },
 {
 field: 'duration', headerText: 'Duration', textAlign:
'Right', width: 80, editType: 'numericedit', edit: { params: { format:
'n'}}
 }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>EJ2 Tree Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Tree Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<!-- Syncfusion Essential JS 2 Styles -->
<link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Cell editing in EJ2 JavaScript Treegrid control

In Cell edit mode, when you double click on a cell, it is changed to edit state.

You can change the cell value and save to the data source.

To enable Cell edit, set the [editSettings.mode](#) as `Cell`.

### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Cell' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {

```

```

 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', editType: 'datepickeredit', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
 'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

Cell edit mode is default mode of editing.

### Row editing in EJ2 JavaScript Treegrid control

In Row edit mode, when you start editing the currently selected record, the entire row is changed to edit state.

You can change the cell values of the row and save edited data to the data source.

To enable Row edit, set the [editSettings.mode](#) as **Row**.



**INDEX.TS**

```
import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Row' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', editType: 'datepickeredit', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
```

```

if(ele) {
 ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Dialog editing in EJ2 JavaScript Treegrid control

In Dialog edit mode, when you start editing the currently selected row, data will be shown on a dialog.

You can change the cell values and save edited data to the data source.

To enable Dialog edit, set the [editSettings.mode](#) as **Dialog**.

### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 editType: 'datepickeredit', textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Batch editing in EJ2 JavaScript Treegrid control

In Batch edit mode, when you double-click on the Tree Grid cell, then the target cell changed to edit state. You can bulk save (added, changed and deleted data in the single request) to data source by click on the toolbar's **Update** button or by externally invoking the [batchSave](#) method. To enable Batch edit, set the [editSettings.mode](#) as **Batch**.

### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { projectData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: 'TaskID',
 parentIdMapping: 'parentID',
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Batch', newRowPosition: 'Below' },
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 {
 field: 'StartDate', headerText: 'Start Date', width: 90,
 editType: 'datepickeredit', textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');

```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Template editing in EJ2 JavaScript Treegrid control

*Dialog template*

The dialog template editing provides an option to customize the default behavior of dialog editing. Using the dialog template, you can render your own editors by defining the [editSettings.mode](#) as **Dialog** and [template](#) as SCRIPT element ID or HTML string which holds the template.

In some cases, you need to add the new field editors in the dialog which are not present in the column model. In that situation, the dialog template will help you to customize the default edit dialog.

In the following sample, treegrid enabled with dialog template editing.

**INDEX.TS**

```

import { TreeGrid, Edit, Toolbar, DialogEditEventArgs, SaveEventArgs } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
import { DataUtil } from '@syncfusion/ej2-data';
import { DatePicker } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { CheckBox } from '@syncfusion/ej2-buttons';

```

```

import { NumericTextBox } from '@syncfusion/ej2-inputs';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Dialog',
 template: '#dialogtemplate' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right'},
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'progress', headerText: 'Progress', width: 80, textAlign:
'Right' }
],
 height: 270,
 actionBegin: (args: SaveEventArgs) => {
 if (args.requestType === 'save') {
 // cast string to integer value.
 args.data['progress'] =
parseFloat(args.form.querySelector("#progress").value);
 }
 },
 actionComplete: (args: DialogEditEventArgs) => {
 if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
 // Add Validation Rules
 args.form.ej2_instances[0].addRules('progress', {max: 100});
 // EJ2-control Rendering
 let priorityData: {}[] =
DataUtil.distinct(treeGridObj.grid.dataSource, 'priority', true);
 new DropDownList({value: args.rowData.priority, popupHeight:
'200px', floatLabelType: 'Always',
 dataSource: priorityData, fields: {text: 'priority', value:
'priority'}, placeholder: 'Priority'},
args.form.elements.namedItem('priority') as HTMLInputElement);
 new DatePicker({value: args.rowData.startDate, floatLabelType:
'Always', placeholder: 'Start Date'},
args.form.elements.namedItem('startDate') as HTMLInputElement)
 new DatePicker({value: args.rowData.endDate, floatLabelType:
'Always', placeholder: 'End Date'}, args.form.elements.namedItem('endDate')
as HTMLInputElement)
 new CheckBox({ label: 'Approved', checked: args.rowData.approved
}, args.form.elements.namedItem('approved'));
 new NumericTextBox({value: args.rowData.progress, format: 'n',
placeholder: 'Progress', floatLabelType: 'Always' },
args.form.elements.namedItem('progress') as HTMLInputElement);
 // Set initail Focus
 if (args.requestType === 'beginEdit') {
 (args.form.elements.namedItem('taskName') as
HTMLInputElement).focus();
 }
 }
 }
});

```

```

 }
 }
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.2/css/bootstrap.min.c
ss">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">

```

```

 <div id="TreeGrid"></div>
 </div>
 <script id="dialogtemplate" type="text/x-template">
 <div>
 <div class="form-row" >
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input id="taskID" name="taskID" type="text"
value=${if(isAdd)} '' ${else} ${taskID} ${/if} ${if(isAdd)} '' ${else}
disabled ${/if}/>

 <label class="e-float-text e-label-top"
for="OrderID">Task ID</label>
 </div>
 </div>
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input id="taskName" name="taskName" type="text"
value=${if(isAdd)} '' ${else} ${taskName} ${/if} />

 <label class="e-float-text e-label-top"
for="taskName">Task Name</label>
 </div>
 </div>
 </div>
 <div class="form-row">
 <div class="form-group col-md-6">
 <div>
 <input id="priority" name="priority" type="text"
value=${if(isAdd)} '' ${else} ${priority} ${/if} />
 </div>
 </div>
 <div class="form-group col-md-6">
 <input id="progress" value=${if(isAdd)} '' ${else}
${progress} ${/if} />
 </div>
 </div>
 <div class="form-row">
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input type="text" name="startDate" id="startDate"
value=${if(isAdd)} '' ${else} ${startDate} ${/if} />

 </div>
 </div>
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input type="text" name="endDate" id="endDate"
value=${if(isAdd)} '' ${else} ${endDate} ${/if} />

 </div>
 </div>
 </div>
 <div class="form-row">
 <div class="form-group col-md-6">
 <input type="checkbox" name="approved" id="approved"
${if(isAdd)} '' ${else} checked ${/if} />
 </div>
 </div>
 </div>
 </script>

```

```

 </div>
 </div>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The template form editors should have **name** attribute.

#### Template context

The Essential JS2 packages has a built-in template engine. Using the template engine, you can access the row information inside the HTML element and bind the attributes, values, or elements based on this row information.

The following properties will be available at the time of template execution.

Property Name	Usage
---------------	-------

isAdd	A Boolean property; it defines whether the current row should be a new record or not.
-------	---------------------------------------------------------------------------------------

In the following code example, the **OrderID** textbox has been disabled by using the **isAdd** property.

,

// The disabled attributes will be added based on the isAdd property.

```

<input id="taskID" name="taskID" type="text" value=${if(isAdd)} " ${else} ${taskID} ${/if} ${if(isAdd)}"
${else} disabled ${/if}/>

```

,

The following code example illustrates rendering the **taskID** textbox, when a new record is added.

,

```

${if(isAdd)}

```

```

<div class="form-group col-md-6">

```

```

<div class="e-float-input e-control-wrapper">

```

```

<input id="taskID" name="taskID" type="text" value=${if(isAdd)} " ${else} ${taskID} ${/if} ${if(isAdd)}"
${else} disabled ${/if}/>

```

```



```

```

<label class="e-float-text e-label-top" for="OrderID">Task ID</label>

```

```

</div>

```

```

</div>

```



```
$/if}
```

The dialog template syntax supports the ES6 expression string literals, and you can refer to the [Template Engine](#) for more template syntax.

#### Render editors as components

You can convert the form editors to EJ2 controls in the [actionComplete](#) event based on the `requestType` as `beginEdit` or `add`.

The following code example illustrates rendering the drop-down list control in the `actionComplete` event.

```
`ts
actionComplete: (args: DialogEditEventArgs) => {
 if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {
 let priorityData: {}[] = DataUtil.distinct(treeGridObj.grid.dataSource, 'priority', true);
 new DropDownList({value: args.rowData.priority, popupHeight: '200px', floatLabelType: 'Always',
 dataSource: priorityData, fields: {text: 'priority', value: 'priority'}, placeholder: 'Priority'},
 args.form.elements.namedItem('priority') as HTMLInputElement);
 }
}
```

#### Get value from editor

You can read, format, and update the current editor value in the [actionBegin](#) event at the time of setting `requestType` to `save`.

In the following code example, the `progress` value has been formatted and updated.

```
`ts
actionBegin: (args: SaveEventArgs) => {
 if (args.requestType === 'save') {
 // cast string to integer value.
 args.data['progress'] = parseFloat(args.form.querySelector("#progress").value);
 }
}
```

#### Set focus to editor

By default, the first input element in the dialog will be focused while opening the dialog.

If the first input element is in disabled or hidden state, focus the valid input element in the [actionComplete](#) event based on `requestType` as `beginEdit`.

```
`ts
```

```

actionComplete: (args: DialogEditEventArgs) => {
// Set initail Focus
if (args.requestType === 'beginEdit') {
(args.form.elements.namedItem('taskName')as HTMLInputElement).focus();
}
}
,

```

#### Adding validation rules for custom editors

If you have used additional fields that are not present in the column model, then add the validation rules to the [actionComplete](#) event.

```

`ts
actionComplete: (args: DialogEditEventArgs) => {
if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {
// Add Validation Rules
args.form.ej2_instances[0].addRules('progress', {max: 100});
}
}
,

```

#### Command column editing in EJ2 JavaScript Treegrid control

The command column provides an option to add CRUD action buttons in a column. This can be defined by the [column.commands](#) property.

The available built-in command buttons are:

Command Button   Actions
----- -----
Edit   Edit the current row.
Delete   Delete the current row.
Save   Update the edited row.
Cancel   Cancel the edited state.

#### INDEX.TS

```

import { TreeGrid, CommandColumn, Edit } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(CommandColumn, Edit);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Row' },
 treeColumnIndex: 1,

```

```

 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right'},
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 {
 headerText: 'Manage Records', width: 130,
 commands: [{ type: 'Edit', buttonOption: { iconCss: ' e-icons e-
edit', cssClass: 'e-flat' } },
 { type: 'Delete', buttonOption: { iconCss: 'e-icons e-
delete', cssClass: 'e-flat' } },
 { type: 'Save', buttonOption: { iconCss: 'e-icons e-update',
cssClass: 'e-flat' } },
 { type: 'Cancel', buttonOption: { iconCss: 'e-icons e-
cancel-icon', cssClass: 'e-flat' } }]
 }
],
 height: 270
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

### Custom command

The custom command buttons can be added in a column by using the [column.commands](#) property and the action for the custom buttons can be defined in the [buttonOption.click](#) event.

### INDEX.TS

```
import { TreeGrid, CommandColumn, Edit, IRow, Column } from
 '@syncfusion/ej2-treegrid';
import { closest } from '@syncfusion/ej2-base';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(CommandColumn, Edit);
let onClick = (args: Event) => {
 let rowIndex: number = (<HTMLTableRowElement>closest(args.target as
Element, '.e-row')).rowIndex;
 let data: Object = treeGridObj.getCurrentViewRecords();
 alert(JSON.stringify(data[rowIndex]));
}
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Row' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 { headerText: 'Commands', width: 120, commands: [{ buttonOption: {
content: 'Details', cssClass: 'e-flat', click: onClick } }] },
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Validation in EJ2 JavaScript Treegrid control

### Column validation

Column validation allows you to validate the edited or added row data and it display errors for invalid fields before saving data.

TreeGrid uses **Form Validator** component for column validation.

You can set validation rules by defining the [columns.validationRules](#).

### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, showDeleteConfirmDialog: true, mode: 'Cell' },
 treeColumnIndex: 1,
 columns: [
 {
 field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
 textAlign: 'Right',
 validationRules: { required: true, number: true }, width: 100
 },
 {
 field: 'taskName', headerText: 'Task Name', editType:
'stringedit',
 validationRules: { required: true }, width: 170
 },
 {
 field: 'startDate', headerText: 'Start Date', textAlign:
'Right', width: 180,

```

```

 editType: 'datetimepickeredit', edit: { params: { format: 'M/d/y
hh:mm a' } } },
 format: { format: 'M/d/y hh:mm a', type: 'dateTime' },
validationRules: { date: true }
 },
 {
 field: 'approved', headerText: 'Approved', width: 110, editType:
'booleanedit',
 type: 'boolean', displayAsCheckBox: true
 },
 {
 field: 'progress', headerText: 'Progress', textAlign: 'Right',
width: 120,
 editType: 'numericedit', validationRules: { number: true, min: 0
}, edit: { params: { format: 'n' } }
 },
 { field: 'priority', headerText: 'Priority', width: 110, editType:
'dropdownedit', validationRules: { required: true } }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Custom validation

You can define your own custom validation rules for the specific columns by using **Form Validator custom rules**.

In the below demo, custom validation applied for **Priority** column.

### INDEX.TS

```
import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let customFn: (args: { [key: string]: string }) => boolean = (args: { [key:
string]: string }) => {
 return args['value'].length <= 8;
};
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Add', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true, showDeleteConfirmDialog: true, mode: 'Cell' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', editType: 'datepickeredit', type: 'date', format: 'yMd'
 },
 { field: 'priority', headerText: 'Priority', width: 80,
 validationRules: { required: true, minLength: [customFn, 'Value should be
within 8 letters'] } }
],
 height: 270
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Persisting data in server in EJ2 JavaScript Treegrid control

Edited data can be persisted in the database using the RESTful web services.

All the CRUD operations in the treegrid are done through [DataManager](#). The **DataManager** has an option to bind all the CRUD related data in server-side.

For your information, the ODataAdaptor persists data in the server as per OData protocol.

In the following section, we have explained how to perform CRUD operation in server-side using the [UrlAdaptor](#) and **RemoteSave Adaptor**.

#### *URL adaptor*

You can use the [UrlAdaptor](#) of **DataManager** when binding data source from remote data.

In the initial load of treegrid, data are fetched from remote data and bound to the treegrid using **url** property of **DataManager**.

You can map The CRUD operation in treegrid can be mapped to server-side Controller actions using the properties **insertUrl**, **removeUrl**, **updateUrl** and **batchUrl**.

The following code example describes the above behavior.

Also, when using the **UrlAdaptor**, you need to return the data as JSON from the controller action and the JSON object must contain a property as **result** with **dataSource** as its value and one more property **count** with the **dataSource** total records count as its value.

The following code example describes the above behavior.

```

`ts
public ActionResult DataSource(DataManager dm)
{
 var DataSource = TreeData.GetTree();

```



```

DataOperations operation = new DataOperations();
if (dm.Where != null && dm.Where.Count > 0)
{
 DataSource = operation.PerformFiltering(DataSource, dm.Where, "and"); //perform filtering and
 maintain child records on Expand/Collapse operation
}
var count = DataSource.ToList<TreeData>().Count();
if (dm.Skip != 0)
{
 DataSource = operation.PerformSkip(DataSource, dm.Skip); //Paging
}
if (dm.Take != 0)
{
 DataSource = operation.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}
`

```

#### *Insert record*

Using the `insertUrl` property, you can specify the controller action mapping URL to perform insert operation on the server-side.

The following code example describes the above behavior and also we have inserted new record based on the `newRowPosition` TreeGrid editSettings as "Below".

```

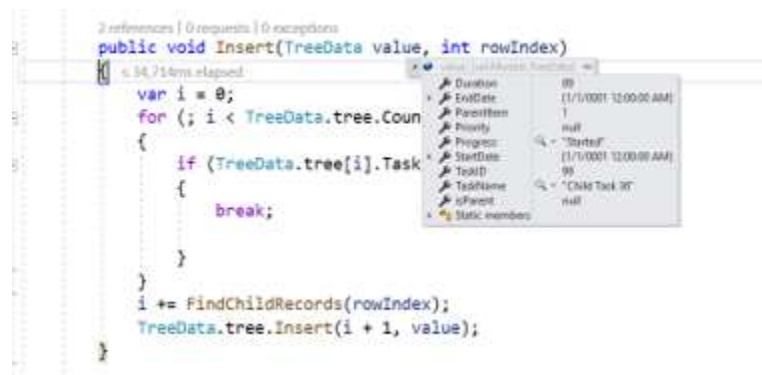
`ts
public void Insert(TreeGridData value, int relationalKey)
{
 var i = 0;
 for (; i < TreeData.tree.Count; i++)
 {
 if (TreeData.tree[i].TaskID == relationalKey)
 {
 break;
 }
 }
}

```

`i += FindChildRecords(relationalKey);` // Inserted new record when newRowPosition API is in "Below".

```
TreeData.tree.Insert(i + 1, value);
}
public int FindChildRecords(int id)
{
 var count = 0;
 for (var i = 0; i < TreeData.tree.Count; i++)
 {
 if (TreeData.tree[i].ParentItem == id)
 {
 count++;
 count += FindChildRecords(TreeData.tree[i].TaskID);
 }
 }
 return count;
}
```

The newly added record details are bound to the `value` parameter and `relationalKey` contains primaryKey value of an selected record helps to find out the position of newly added record. Please refer to the following screenshot.



#### Update record

Using the `updateUrl` property, the controller action mapping URL can be specified to perform save/update operation on the server-side.

The following code example describes the previous behavior.

```
`ts
public ActionResult Update(TreeGridData value)
{

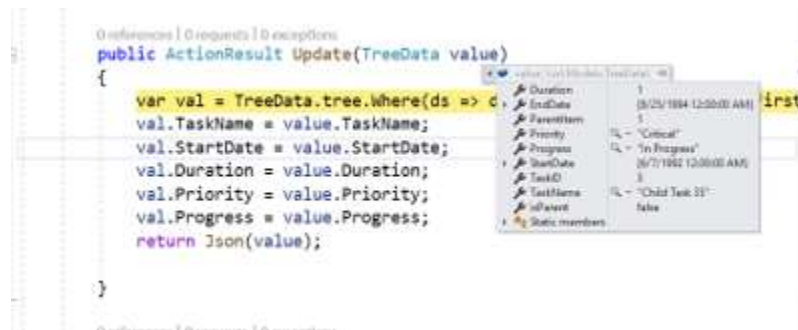
```

```

var val = TreeData.tree.Where(ds => ds.TaskID == value.TaskID).FirstOrDefault();
val.TaskName = value.TaskName;
val.StartDate = value.StartDate;
val.Duration = value.Duration;
val.Priority = value.Priority;
val.Progress = value.Progress;
return Json(value);
}

```

The updated record details are bound to the `value` parameter. Please refer to the following screenshot.



#### Delete record

Using the `removeUrl` and `batchUrl` property, the controller action mapping URL can be specified to perform delete operation on the server-side.

The following code example describes the previous behavior.

```

`ts
public ActionResult Delete(int key)
{
 TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == key).FirstOrDefault());
}

// Remove method (batchUrl) will be triggered when we delete parent record.
public ActionResult Remove(List<TreeGridData> changed, List<TreeGridData> added,
 List<TreeGridData> deleted)
{
 for (var i = 0; i < deleted.Count; i++)
 {
 TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == deleted[i].TaskID).FirstOrDefault());
 }
}

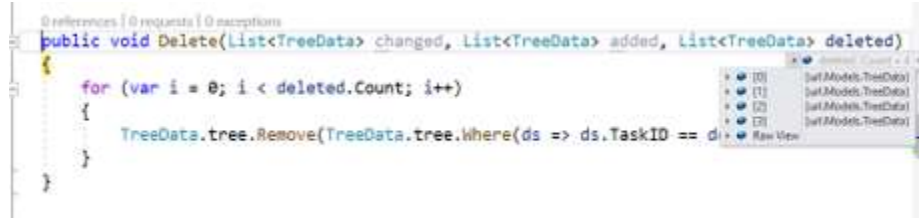
```

```

}
,

```

The deleted record primary key value is bound to the **key** parameter. Please refer to the following screenshot.



While delete parent record, the parent and child records is bound to the **deleted** parameter. Please refer to the following screenshot.



#### Remote save adaptor

You may need to perform all Tree Grid Actions in client-side except the CRUD operations, that should be interacted with server-side to persist data. It can be achieved in TreeGrid by using **RemoteSaveAdaptor**.

Datasource must be set to **json** property and set **RemoteSaveAdaptor** to the **adaptor** property. CRUD operations can be mapped to server-side using **updateUrl**, **insertUrl**, **removeUrl** and **batchUrl** properties.

You can use the following code example to use **RemoteSaveAdaptor** in TreeGrid.

The following code example describes how to fetch the data from **ViewBag** in angular.

```

,

```

```

<script type="text/javascript">

```

```

window.griddata = '@Html.Raw(Json.Encode(ViewBag.dataSource))';

```

```

</script>

```

```

,

```

The following code example describes the CRUD operations handled at server-side.

```

`ts

```

```

public ActionResult Index(DataManager dm)

```

```

{

```

```

var data = TreeData.GetTree();

```

```

ViewBag.dataSource = data;

```

```
return View();
}
public void Insert(TreeData value, int relationalKey)
{
 var i = 0;
 for (; i < TreeData.tree.Count; i++)
 {
 if (TreeData.tree[i].TaskID == relationalKey)
 {
 break;
 }
 }
 i += FindChildRecords(relationalKey); // Inserted new record when newRowPosition API is in "Below".
 TreeData.tree.Insert(i + 1, value);
}
public ActionResult Update(TreeData value)
{
 var val = TreeData.tree.Where(ds => ds.TaskID == value.TaskID).FirstOrDefault();
 val.TaskName = value.TaskName;
 val.StartDate = value.StartDate;
 val.Duration = value.Duration;
 val.Priority = value.Priority;
 val.Progress = value.Progress;
 return Json(value);
}
public ActionResult Delete(int key)
{
 TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == key).FirstOrDefault());
}
// Remove method (batchUrl) will be triggered when we delete parent record.
public ActionResult Remove(List<TreeGridData> changed, List<TreeGridData> added,
List<TreeGridData> deleted)
{

```

```

for (var i = 0; i < deleted.Count; i++)
{
 TreeData.tree.Remove(TreeData.tree.Where(ds => ds.TaskID == deleted[i].TaskID).FirstOrDefault());
}
}
,

```

### Sorting in EJ2 JavaScript Treegrid control

Sorting enables you to sort data in the **Ascending** or **Descending** order.

To sort a column, click the column header.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the TreeGrid, set the [allowSorting](#) to true. Sorting options can be configured through the [sortSettings](#).

To sort, inject the [Sort](#) module in the treegrid.

#### INDEX.TS

```

import { TreeGrid, Sort } from '@syncfusion/ej2-treegrid';
import { sortData } from './datasource.ts';
TreeGrid.Inject(Sort);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sortData,
 childMapping: 'subtasks',
 allowSorting: true,
 height: 315,
 columns: [
 { field: 'Category', headerText: 'Category', width: 140 },
 { field: 'orderName', headerText: 'Order Name', width: 200 },
 { field: 'orderDate', headerText: 'Order Date', width: 150,
 textAlign: 'Right', format: 'yMd', type: 'date' },
 { field: 'units', headerText: 'Units', width: 90, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* TreeGrid columns are sorted in the **Ascending** order. If you click the already sorted column, the sort direction toggles.

\* You can apply and clear sorting by invoking [sortByColumn](#) and [clearSorting](#) methods.

\* To disable sorting for a particular column, set the [columns.allowSorting](#) to false.

#### Initial sort

To sort at initial rendering, set the [field](#) and [direction](#) in the **sortSettings.columns**.

**INDEX.TS**

```
import { TreeGrid, Sort } from '@syncfusion/ej2-treegrid';
import { sortData } from './datasource.ts';
TreeGrid.Inject(Sort);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sortData,
 childMapping: 'subtasks',
 allowSorting: true,
 height: 315,
 treeColumnIndex: 1,
 sortSettings: { columns: [{ field: 'Category', direction: 'Ascending' },
 { field: 'orderName', direction: 'Ascending' }] },
 columns: [
 { field: 'Category', headerText: 'Category', width: 140 },
 { field: 'orderName', headerText: 'Order Name', width: 200 },
 { field: 'orderDate', headerText: 'Order Date', width: 120,
 textAlign: 'Right', format: 'yMd', type: 'date' },
 { field: 'units', headerText: 'Units', width: 90, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Sorting events

During the sort action, the treegrid component triggers two events. The [actionBegin](#) event triggers before the sort action starts, and the [actionComplete](#) event triggers after the sort action is completed. Using these events you can perform the needed actions.

### INDEX.TS

```

import { TreeGrid, Sort } from '@syncfusion/ej2-treegrid';
import { sortData } from './datasource.ts';
TreeGrid.Inject(Sort);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sortData,
 childMapping: 'subtasks',
 allowSorting: true,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'Category', headerText: 'Category', width: 140 },
 { field: 'orderName', headerText: 'Order Name', width: 200 },
 { field: 'orderDate', headerText: 'Order Date', width: 120,
 textAlign: 'Right', format: 'yMd', type: 'date' },
 { field: 'units', headerText: 'Units', width: 90, textAlign: 'Right'
 }
],
 actionBegin: actionHandler,
 actionComplete: actionHandler
});
treeGridObj.appendTo('#TreeGrid');
function actionHandler(args: SortEventArgs) {
 alert(args.requestType + ' ' + args.type); //custom Action
}

```

```
}

```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The `args.requestType` is the current action name. For example, in sorting the `args.requestType` value is 'sorting'.

<!-- Custom sort comparer

You can customize the default sort action for a column by defining the [column.sortComparer](#) property. The sort comparer function has the same functionality like [Array.sort](#) sort comparer.

In the following example, custom sort comparer function was defined in the `Category` column.

### INDEX.TS

```

import { TreeGrid, Sort } from '@syncfusion/ej2-treegrid';
import { sortData } from './datasource.ts';
TreeGrid.Inject(Sort);
// The custom function
let sortComparer: (reference: string, comparer: string) => number =
(reference: string,
comparer: string) => {
 if (reference < comparer) {
 return -1;
 }
 if (reference > comparer) {
 return 1;
 }
 return 0;
};
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sortData,
 childMapping: 'subtasks',
 allowSorting: true,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'Category', headerText: 'Category', width: 140 },
 { field: 'orderName', headerText: 'Order Name', width: 200 },
 { field: 'orderDate', headerText: 'Order Date', width: 120,
textAlign: 'Right', format: 'yMd', type: 'date' },
 { field: 'units', headerText: 'Units', width: 90, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The sort comparer function will work only for the local data. -->

### Touch interaction

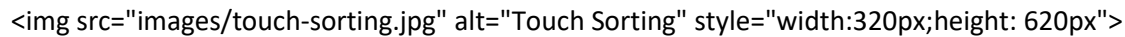
When you tap the treegrid header on touchscreen devices, the selected column header is sorted. A



popup is displayed for multi-column sorting. To sort multiple columns, tap the popup, and then tap the desired treegrid headers.



The following screenshot shows treegrid touch sorting.

 Touch Sorting" style="width:320px;height: 620px">

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Filtering

#### Filtering in EJ2 JavaScript Treegrid control

Filtering allows you to view specific or related records based on filter criteria. To enable filtering in the TreeGrid, set the [allowFiltering](#) to true. Filtering options can be configured through [filterSettings](#).

To use filter, inject the `Filter` module in the treegrid.

#### INDEX.TS

```
import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 allowFiltering: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* You can apply and clear filtering by using [filterByColumn](#) and [clearFiltering](#) methods.

\* To disable filtering for a particular column, set

[columns.allowFiltering](#) to false.

*Filter hierarchy modes*

TreeGrid provides support for a set of filtering modes with [filterSettings.filterHierarchyMode](#) property.

The below are the type of filter mode available in TreeGrid.

- **Parent** : This is the default filter hierarchy mode in TreeGrid. The filtered records are displayed with its parent records, if the filtered records not have any parent record then the filtered records only displayed.
- **Child** : The filtered records are displayed with its child record, if the filtered records not have any child record then the filtered records only displayed.
- **Both** : The filtered records are displayed with its both parent and child record, if the filtered records not have any parent and child record then the filtered records only displayed.
- **None** : The filtered records are only displayed.

**INDEX.TS**

```
import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 225,
 allowFiltering: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
let dropDownMode: DropDownList = new DropDownList({
 dataSource: [
 { id: 'Parent', mode: 'Parent' },
 { id: 'Child', mode: 'Child' },
 { id: 'Both', mode: 'Both' },
 { id: 'None', mode: 'None' },
],
 fields: { text: 'mode', value: 'id' },
 value: 'Parent',
 change: (e: ChangeEventArgs) => {
 let mode: any = <string>e.value;
 treeGridObj.filterSettings.hierarchyMode = mode;
 treeGridObj.clearFiltering();
 }
});
```

```
dropDownMode.appendTo('#mode');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div>
 <div style="padding-top: 7px; display: inline-block">Hierarchy
Mode</div>
 <div style="display: inline-block">
 <input type="text" id="mode">
 </div>
 </div>
```



```

 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Initial filter

To apply the filter at initial rendering, set the filter `predicate` object in

[filterSettings.columns](#).

### INDEX.TS

```

import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 allowFiltering: true,
 filterSettings: {
 columns: [{ field: 'taskName', matchCase: false, operator:
'startswith', predicate: 'and', value: 'plan' },
 { field: 'duration', matchCase: false, operator: 'equal', predicate:
'and', value: 5 }]
 },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Filter operators

The filter operator for a column can be defined in the `filterSettings.columns.operator` property.

The available operators and its supported data types are:

Operator	Description	Supported Types
----------	-------------	-----------------

startswith | Checks whether the value begins with the specified value. |String

endswith | Checks whether the value ends with the specified value. |String

contains | Checks whether the value contains the specified value. |String

equal | Checks whether the value is equal to the specified value. |String &#124; Number &#124; Boolean &#124; Date

notequal | Checks for values not equal to the specified value. |String &#124; Number &#124; Boolean &#124; Date

greaterthan | Checks whether the value is greater than the specified value. |Number &#124; Date

greaterthanorequal | Checks whether a value is greater than or equal to the specified value. |Number &#124; Date

lessthan | Checks whether the value is less than the specified value. |Number &#124; Date

lessthanorequal | Checks whether the value is less than or equal to the specified value. |Number &#124; Date

By default, the `filterSettings.columns.operator` value is `equal`.

#### *Diacritics*

By default, treegrid ignores diacritic characters while filtering. To include diacritic characters, set the `filterSettings.ignoreAccent` as `true`.

In the following sample, type **aero** in **Name** column to filter diacritic characters.

#### **INDEX.TS**

```
import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { diacritics } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: diacritics,
 childMapping: 'Children',
 height: 275,
 allowFiltering: true,
 filterSettings: { ignoreAccent: true },
 treeColumnIndex: 0,
 columns: [
 { field: 'EmpID', headerText: 'Employee ID', width: 95 },
 { field: 'Name', headerText: 'Name', width: 110 },
 { field: 'DOB', headerText: 'DOB', width: 90, textAlign: 'Right',
format: 'yMd' },
 { field: 'Country', width: 65 }
]
});
treeGridObj.appendTo('#TreeGrid');
```

#### **INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Filter bar in EJ2 JavaScript Treegrid control

By setting the [allowFiltering](#) to true, the filter bar row will render next to the header, which allows you to filter data. You can filter the records with different expressions depending upon the column type.

#### Filter bar expressions:

You can enter the following filter expressions (operators) manually in the filter bar.

Expression	Example	Description	Column Type
=	=value	equal	Number
!=	!=value	notequal	Number
>	>value	greaterthan	Number
<	<value	lessthan	Number
=	>=value	greaterthanorequal	Number
<=	<=value	lessthanorequal	Number
/	/value	startswith	String
%	%value	endswith	String
N/A	N/A	Equal	Date
N/A	N/A	Equal	Boolean

#### INDEX.TS

```
import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 allowFiltering: true,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

*Filter bar template with custom component*

The [filterBarTemplate](#) is used to add custom components to a particular column, and does the following functions:

- **create:** Creates custom components.
- **write:** Wires events for custom components.

In the following sample, the dropdown is used as a custom component in the Duration column.

**INDEX.TS**

```
import { TreeGrid, Filter, Column } from '@syncfusion/ej2-treegrid';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 allowFiltering: true,
 filterSettings: { type: 'FilterBar', hierarchyMode: 'Parent', mode:
'Immediate' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right',
 filterBarTemplate: {
 create: (args: { element: Element, column: Column }) => {
 let dd: HTMLInputElement =
document.createElement('input');
 dd.id = 'duration';
 return dd;
 },
 write: (args: { element: Element, column: Column }) => {
 let dataSource: string[] = ['All', '1', '3', '4', '5',
'6', '8', '9'];
 this.dropDownFilter = new DropDownList({
 dataSource: dataSource,
 value: 'All',
 change: (e: ChangeEventArgs) => {
 let valuenum: any = +e.value;
 let id: any =
<string>this.dropDownFilter.element.id;
 let value: any = <string>e.value;
 if (value !== 'All') {
 treeGridObj.filterByColumn(id, 'equal',
valuenum);
 } else {
```

```

treeGridObj.removeFilteredColsByField(id);
 }
 }
 });
 this.dropDownFilter.appendTo('#duration');
 }
 }
]
});
treeGridObj.appendTo('#TreeGrid');

```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

</head>

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
```



```
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Change default filter bar operator

You can change the default filter operator by extending [filterModule.filterOperators](#) property in [dataBound](#) event.

In the following sample, we have changed the default operator for string typed columns as **contains** from **startsWith**.

### INDEX.TS

```
import { TreeGrid, Filter, Column } from '@syncfusion/ej2-treegrid';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 dataBound: dataBound,
 allowFiltering: true,
 filterSettings: { type: 'FilterBar', hierarchyMode: 'Parent', mode:
'Immediate' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
function dataBound(): void {
 Object.assign(treeGridObj.grid.filterModule.filterOperators, {
 startsWith: 'contains' });
}
```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Filter menu in EJ2 JavaScript Treegrid control

You can enable filter menu by setting the [filterSettings.type](#) as **Menu**. The filter menu UI will be rendered based on its column type, which allows you to filter data.

You can filter the records with different operators.

#### INDEX.TS

```
import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 allowFiltering: true,
 filterSettings: {type: 'Menu'},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 120, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 }
]
});
treeGridObj.appendTo('#TreeGrid');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* [allowFiltering](#) must be set as true to enable filter menu.

\* Setting [columns.allowFiltering](#) as false will prevent filter menu rendering for a particular column.

#### *Custom component in filter menu*

The [column.filter.ui](#) is used to add custom filter components to a particular column. To implement custom filter ui, define the following functions:

- **create**: Creates custom component.
- **write**: Wire events for custom component.
- **read**: Read the filter value from custom component.

In the following sample, dropdown is used as custom component in the duration column.

#### **INDEX.TS**

```

import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
import { DataManager } from '@syncfusion/ej2-data';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
TreeGrid.Inject(Filter);

```

```

let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 275,
 allowFiltering: true,
 filterSettings: {type: 'Menu'},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 120,
textAlign: 'Left' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right', filter: {
 ui: {
 create: (args: { target: Element, column: Object }) => {
 let flValInput: HTMLElement = createElement('input',
{ className: 'flm-input' });
 args.target.appendChild(flValInput);
 let dataSource: string[] = ['All', '1', '3', '4',
'5', '6', '8', '9'];
 this.dropDownFilter = new DropDownList({
 dataSource: dataSource,
 value: 'All', popupHeight: '200px'
 });
 this.dropDownFilter.appendTo(flValInput);
 },
 write: (args: {
 column: Object, target: Element, parent: any,
 filteredValue: number | string
 }) => {
 this.dropDownFilter.value = args.filteredValue;
 },
 read: (args: { target: Element, column: any, operator:
string, fltrObj: Filter }) => {
 args.fltrObj.filterByColumn(args.column.field,
args.operator, parseInt(this.dropDownFilter.value));
 }
 }
 }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### *Enable different filter dialog for a column*

You can use both **Menu** and **Excel** filter in a same TreeGrid. To do so, set the [column.filter.type](#) as **Menu** or **Excel**.

In the following sample menu filter is enabled by default and excel filter is enabled for the Task Name column using the [column.filter.type](#).

### **INDEX.TS**

```
import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 273,
 allowFiltering: true,
 filterSettings: { type: 'Menu' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150, textAlign:
'Left', filter: { type: 'Excel' } },
 { field: 'duration', headerText: 'Duration', width: 90, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 90, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Excel like filter in EJ2 JavaScript Treegrid control

You can enable Excel like filter by defining [filterSettings.type](#) as **Excel**. The excel menu contains an option such as Sorting, Clear filter, Sub menu for advanced filtering.

#### INDEX.TS

```

import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 273,
 allowFiltering: true,
 filterSettings: { type: 'Excel' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150, textAlign:
'Left' },
 { field: 'duration', headerText: 'Duration', width: 90, textAlign:
'Right' },
 { field: 'progress', headerText: 'Progress', width: 90, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```



**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

*Change default excel filter operator*

You can change the default excel-filter operator by changing the column operator as `contains` from `startsWith` in the [actionBegin](#) event

**INDEX.TS**

```
import { TreeGrid, Filter, Column } from '@syncfusion/ej2-treegrid';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 actionBegin: actionBegin,
 allowFiltering: true,
 filterSettings: { type: 'Excel' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 75, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
function actionBegin(e) {
 if(e.requestType === 'filtersearchbegin' && e.column.type === "string")
 {
 e.operator = 'contains';
 }
}
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Searching in EJ2 JavaScript Treegrid control

You can search records in a TreeGrid, by using the [search](#) method with search key as a parameter. This also provides an option to integrate search text box in treegrid's toolbar by adding [search](#) item to the [toolbar](#).

To search records, inject the [Filter](#) module in the treegrid.

### INDEX.TS

```

import { TreeGrid, Filter, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({

```

```

 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Search'],
 height: 270,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Initial search

To apply search at initial rendering, set the fields, operator, key, and ignoreCase in the [searchSettings](#).

### INDEX.TS

```

import { TreeGrid, Filter, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Search'],
 searchSettings: { fields: ['taskName'], operator: 'contains', key:
'plan', ignoreCase: true },
 height: 270,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, treegrid searches all the bound column values. To customize this behavior define the [searchSettings.fields](#) property.

### Search operators

The search operator can be defined in the [searchSettings.operator](#) property to configure specific searching.

The following operators are supported in searching:

#### Operator | Description

startsWith | Checks whether a value begins with the specified value.

endsWith | Checks whether a value ends with the specified value.

contains | Checks whether a value contains the specified value.

equal | Checks whether a value is equal to the specified value.

notEqual | Checks for values not equal to the specified value.

By default, the [searchSettings.operator](#) value is **contains**.

### Search by external button

To search treegrid records from an external button, invoke the [search](#) method.

#### INDEX.TS

```
import { TreeGrid, Filter } from '@syncfusion/ej2-treegrid';
import { Button } from '@syncfusion/ej2-buttons';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 220,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
let searchBtn: Button = new Button();
searchBtn.appendTo('#search');
document.getElementById('search').addEventListener('click', () => {
 let searchText: string =
 (<HTMLInputElement>document.getElementsByClassName('searchtext')[0]).value;
 treeGridObj.search(searchText);
});
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="e-float-input" style="width: 200px; display: inline-
block;">
 <input type="text" class="searchtext">

 <label class="e-float-text">Search text</label>
 </div>
 <button id="search">Search</button>
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>

```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Search specific columns

By default, treegrid searches all visible columns. You can search specific columns by defining the specific column's field names in the [searchSettings.fields](#) property.

### INDEX.TS

```
import { TreeGrid, Filter, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter, Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Search'],
 searchSettings: { fields: ['taskName', 'duration'] },
 height: 270,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

## Selection

### Selection in EJ2 JavaScript Treegrid control

Selection provides an option to highlight a row or a cell. It can be done through simple mouse down or arrow keys. To disable selection in TreeGrid, set the [allowSelection](#) to false.

The treegrid supports two types of selection that can be set by using the [selectionSettings.type](#). They are:

- **Single:** The `Single` value is set by default, and it only allows selection of a single row or a cell.
- **Multiple:** Allows you to select multiple rows or cells.

To perform the multi-selection, press and hold CTRL key and click the desired rows or cells. To select range of rows or cells, press and hold the SHIFT key and click the rows or cells.

## INDEX.TS

```
import { TreeGrid, Selection } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 selectionSettings: { type: 'Multiple' },
 height: 270,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">
```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Selection mode

TreeGrid supports three types of selection mode which can be set by using

[selectionSettings.mode](#). They are:

- **Row** - The **row** value is set by default. Allows you to select rows only.
- **Cell** - Allows you to select cells only.
- **Both** - Allows you to select rows and cells at the same time.

### INDEX.TS

```

import { TreeGrid, Selection } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 selectionSettings: { mode: 'Both' },
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 { field: 'progress', headerText: 'progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

*Touch interaction*

When you tap a treegrid row on touchscreen device, the tapped row is selected.



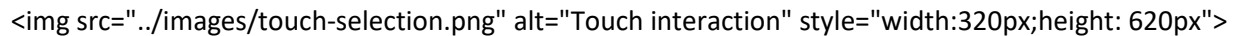
It also shows a popup for multi-row selection.



To select multiple rows or cells, tap the popup and then tap the desired rows or cells.

Multi-selection requires the selection [type](#) to be **multiple**.

The following screenshot represents a treegrid touch selection in the device.

 Touch interaction" style="width:320px;height: 620px">

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Row selection in EJ2 JavaScript Treegrid control

#### *Select row at initial rendering*

To select a row at initial rendering, set the [selectedRowIndex](#) value.

#### **INDEX.TS**

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 270,
 treeColumnIndex: 1,
 selectionSettings: { type: 'Multiple', mode: 'Both' },
 selectedRowIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

#### **INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Get selected row indexes

You can get the selected row indexes by using the [getSelectedRowIndexes](#) method.

### INDEX.TS

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
```

```

import { RowSelectEventArgs } from '@syncfusion/ej2-grids';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 270,
 treeColumnIndex: 1,
 selectionSettings: {type: 'Multiple'},
 rowSelected: rowSelected
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' },
 { field: 'progress', headerText: 'progress', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
function rowSelected(args: RowSelectEventArgs) {
 let selectedrowindex: number[] = treeGridObj.getSelectedRowIndexes();
 // get the selected row indexes.
 alert(selectedrowindex); // to alert the selected row indexes.
 let selectedrecords: Object[] = treeGridObj.getSelectedRecords(); //
 get the selected records.
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Multiple selection based on condition

You can select multiple treegrid rows based on condition by using the [selectRows](#) method.

In the following code, the rows which contains `taskID` value as `3` and `5` are selected at initial rendering.

### INDEX.TS

```

import { TreeGrid, Row, Column } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 270,
 treeColumnIndex: 1,
 selectionSettings: {type: 'Multiple'},
 dataBound: ()=>{
 let rowIndexes : number[]=[];
 treeGridObj.grid.dataSource.forEach((data,index)=>{
 if (data.taskID === 3 || data.taskID === 5){
 rowIndexes.push(index);
 }
 });
 treeGridObj.selectRows(rowIndexes);
 },
 columns: [

```

```

 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
 'Right' },
 { field: 'progress', headerText: 'progress', width: 80, textAlign:
 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row or cell. To [enable toggle](#) selection, set enableToggle property of the selectionSettings as true. If you click on the selected row or cell then it will be unselected and vice versa.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 270,
 treeColumnIndex: 1,
 selectionSettings: {enableToggle: true},
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

### Cell selection in EJ2 JavaScript Treegrid control

Cell Selection can be done through simple Mouse down or Arrow keys(up, down, left and right).

TreeGrid supports two types of cell selection mode which can be set by using

[selectionSettings.cellSelectionMode](#). They are:

- **Flow** - The **Flow** value is set by default.

Select range of cells between the start index and end index which includes in between cells of rows.

- **Box** - Select range of cells within the start and end column indexes which includes in between cells of rows within the range.

### INDEX.TS

```
import { TreeGrid, Selection } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 selectionSettings: { cellSelectionMode: 'Box', type: 'Multiple', mode:
'Cell' },
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Check box selection in EJ2 JavaScript Treegrid control

Checkbox selection provides an option to select multiple treegrid records with help of checkbox in each row.

To render the checkbox in each treegrid row, you need to use checkbox column with type as **checkbox** using the column [type](#) property.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 315,
 treeColumnIndex: 2,
 columns: [
 { type: 'checkbox', width: 60 },
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 150 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
textAlign: 'Right' }
]
});

```

```

]
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {

```

```

 ele.style.visibility = "visible";
 }
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* By default, selection is allowed by clicking a treegrid row or checkbox in that row. To allow selection only through checkbox, you can set the

[selectionSettings.checkboxOnly](#) property to true.

\* Selection can be persisted in all the operations using the [selectionSettings.persistSelection](#) property. For persisting selection on the treegrid, any one of the columns should be defined as a primary key using the [columns.isPrimaryKey](#) property.

#### Checkbox selection mode

In checkbox selection, selection can also be done by clicking on rows. This selection provides two types of Checkbox Selection mode which can be set by using the following API, [selectionSettings.checkboxMode](#). The modes are

- **Default:** This is the default value of the checkboxMode. In this mode, user can select multiple rows by clicking rows one by one.
- **ResetOnRowClick:** In ResetOnRowClick mode, when user clicks on a row it will reset previously selected row. Also you can perform multiple-selection in this mode by press and hold CTRL key and click the desired rows. To select range of rows, press and hold the SHIFT key and click the rows.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 270,
 treeColumnIndex: 2,
 selectionSettings: {checkboxMode: 'ResetOnRowClick'},
 columns: [
 { type: 'checkbox', width: 60 },
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'progress', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```



```

<title>EJ2 Grid</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Checkbox Selection feature is intended for row selection only; it is not compatible with cell selection mode.

### Paging in EJ2 JavaScript Treegrid control

Paging provides an option to display TreeGrid data in page segments. To enable paging, set the [allowPaging](#) to true. When paging is enabled, pager component renders at the bottom of the treegrid.

Paging options can be configured through the [pageSettings](#).

To use paging, inject the [Page](#) module in the treegrid.

#### INDEX.TS

```
import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 pageSettings: {pageSize: 7},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 160 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can achieve better performance by using treegrid paging to fetch only a pre-defined number of records from the data source.

### Page Size Mode

Two behaviors are available in TreeGrid paging to display certain number of records in a current page. Following are the two types of [pageSizeMode](#).

- **All** : This is the default mode. The number of records in a page is based on [pageSize](#) property.
- **Root** : The number of root nodes or the 0th level records to be displayed per page is based on [pageSize](#) property.

With [pageSizeMode](#) property as **Root**, only the root level or the 0th level records are considered in records count.

### INDEX.TS

```
import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
```

```
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 height: 265,
 pageSettings: {pageSize: 2, pageSizeMode: 'Root'},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 160 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Template

You can use custom elements inside the pager instead of default elements.

The custom elements can be defined by using the [template](#) property.

Inside this template, you can access the [currentPage](#), [pageSize](#), [pageCount](#), [totalPage](#) and [totalRecordCount](#) values.

### INDEX.TS

```

import { TreeGrid, Page, PageEventArgs } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
TreeGrid.Inject(Page);
let flag: boolean = true;
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 pageSettings: { pageSize: 6, template: '#template' },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 160 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
],

```

```

 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
 dataBound: () => {
 if (flag) {
 flag = false;
 updateTemplate();
 }
 },
 actionComplete: (args: PageEventArgs) => {
 if (args.requestType === 'paging') {
 updateTemplate();
 }
 }
});
treeGridObj.appendTo('#TreeGrid');
let updateTemplate: Function = () => {
 let numeric: NumericTextBox;
 this.numeric = new NumericTextBox({
 min: 1,
 max: 3,
 step: 1,
 width: 75,
 format: '###.##',
 change: (args) => {
 let value: number = args.value;
 treeGridObj.goToPage(value);
 }
 });
 this.numeric.appendTo('#currentPage');
};
let flag: boolean = true;

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script id="template" type="text/x-template">
 <div class="e-pagertemplate">
 <div class="col-lg-12 control-section">
 <div class="content-wrapper">
 <input id="currentPage" type="text" value=${currentPage}
style="padding-left: 10px; text-align: left">
 </div>
 </div>

 <div id="totalPages" class="e-pagertemplatemessage" style="margin-
top: 5px; margin-left: 30px; border: none; display: inline-block ">
 ${currentPage} of ${totalPages} pages
(${totalRecordsCount} items)
 </div>
 </div>
</script>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Pager with Page Size Dropdown

The pager Dropdown allows you to change the number of records in the TreeGrid dynamically. It can be enabled by defining the [pageSettings.pageSize](#) property as true.

`ts

```
pageSettings: {pageSize: 7, pageSizes: true},
```

```
,
```

Task ID	Task Name	Start Date	End Date	Duration	Progress	Priority
1	▼ Planning	2/3/2017	2/7/2017	5	100	Normal
2	Plan timeline	2/3/2017	2/7/2017	5	100	Normal
3	Plan budget	2/3/2017	2/7/2017	5	100	Low
4	Allocate resources	2/3/2017	2/7/2017	5	100	Critical
5	Planning complete	2/7/2017	2/7/2017	0	0	Low
6	▼ Design	2/10/2017	2/14/2017	3	86	High
7	Software Specification	2/10/2017	2/12/2017	3	60	Normal
8	Develop prototype	2/10/2017	2/12/2017	3	100	Critical
9	Get approval from customer	2/13/2017	2/14/2017	2	100	Low
10	Design Documentation	2/13/2017	2/14/2017	2	100	High
1 2 ... 10 Items per page						1 of 4 pages (36 items)

### How to render Pager at the Top of the TreeGrid

By default, Pager will be rendered at the bottom of the TreeGrid. You can also render the Pager at the top of the TreeGrid by using the `dataBound` event.

#### INDEX.TS

```
import { TreeGrid, Page } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page);
let initialGridLoad: boolean = true;
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 pageSettings: {pageSize: 7, pageSizes: true},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 160 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.dataBound = () =>{
 if (initialGridLoad) {
 initialGridLoad = false;
 var pager = document.getElementsByClassName('e-gridpager');
 var topElement;
 if (treeGridObj.toolbar) {
 topElement = document.getElementsByClassName('e-toolbar');
```



```

 } else {
 topElement = document.getElementsByClassName('e-gridheader');
 }
 topElement[0].before(pager[0]);
}
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

During the paging action, the pager component triggers the below three events.

- \* The **created** event triggers when Pager is created.
- \* The **click** event triggers when the numeric items in the pager is clicked.
- \* The **dropDownChanged** event triggers when pageSize DropDownList value is selected.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Scrolling in EJ2 JavaScript Treegrid control

The scrollbar will be displayed in the treegrid when content exceeds the element [width](#) or [height](#). The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the treegrid exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the treegrid element width.
- The [height](#) and [width](#) are used to set the treegrid height and width, respectively.

The default value for [height](#) and [width](#) is **auto**.

#### Set width and height

To specify the [width](#) and [height](#) of the scroller in the pixel, set the pixel value to a number.

#### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 height: 315,
 width: 400,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width:
120, textAlign: 'Right', type: 'date', format: 'yMd'
 },
],

```

```

 { field: 'duration', headerText: 'Duration', width: 110,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</body>
</html>

```

```

var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Responsive with parent container

Specify the [width](#) and [height](#) as **100%** to make the treegrid element fill its parent container.

Setting the [height](#) to **100%** requires the treegrid parent element to have explicit height.

### INDEX.TS

```

import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 height: '100%',
 width: '100%',
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<style>
 .e-resizable {
 resize: both;
 overflow: auto;
 border: 1px solid red;
 padding: 10px;
 height: 300px;
 min-height: 250px;
 min-width: 250px;
 }
 .e-text{
 font-family: Helvetica, sans-serif;
 font-size: 14px;
 }
</style>
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <p class="e-text"> The parent container can be resizable by dragging the
 bottom-right corner.</p>
 <div id="container" class="e-resizable">
 <div id="TreeGrid" style="height:100%"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Scroll to selected row

You can scroll the treegrid content to the selected row position by using the [rowSelected](#) event.

#### INDEX.TS

```
import { TreeGrid, RowSelectEventArgs } from '@syncfusion/ej2-treegrid';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 height: '270',
 width: '100%',
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 rowSelected: rowSelected,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180 },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
let numeric: NumericTextBox = new NumericTextBox({
 width: 200,
 min: 0,
 showSpinButton: false,
 format: 'N',
 placeholder: 'Enter index to select a row',
 change: onchange
}, '#numeric');
function onchange(): void {
 treeGridObj.selectRow(parseInt(numeric.getText(), 10));
}
function rowSelected(args: RowSelectEventArgs) {
 let rowHeight: number =
treeGridObj.getRows()[treeGridObj.getSelectedRowIndex()][0].scrollHeight;
 treeGridObj.getContent().children[0].scrollTop = rowHeight *
treeGridObj.getSelectedRowIndex()[0];
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input id="numeric" type="text">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Frozen in EJ2 JavaScript Treegrid control

### Frozen rows and columns

Frozen rows and columns provides an option to make rows and columns always visible in the top and left side of the tree grid while scrolling.

To use frozen rows and columns support, inject the **Freeze** module in the tree grid.

In this demo, the [frozenColumns](#) is set as '2' and the [frozenRows](#)

is set as '3'. Hence, the left two columns and top three rows are frozen.

### INDEX.TS

```
import { TreeGrid, Freeze } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Freeze);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 height: 317,
 childMapping: 'subtasks',
 allowSelection: false,
 frozenRows: 3,
 frozenColumns: 2,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign:
'Right', width: 100 },
 { field: 'taskName', headerText: 'Task Name', width: 230 },
 { field: 'startDate', headerText: 'Start Date', width: 120,
textAlign: 'Right',
 type: 'date', format: { type: 'dateTime', format:
'dd/MM/yyyy' } },
 { field: 'endDate', headerText: 'End Date', width: 120,
textAlign: 'Right',
 type: 'date', format: { type: 'dateTime', format:
'dd/MM/yyyy' } },
 { field: 'duration', headerText: 'Duration', textAlign:
'Right', width: 110 },
 { field: 'progress', headerText: 'Progress', textAlign:
'Right', width: 120 },
 { field: 'priority', headerText: 'Priority', textAlign:
'Left', width: 120 },
 { field: 'approved', headerText: 'Approved', width: 110,
textAlign: 'Left' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Freeze particular columns

You can use [isFrozen](#) property to freeze selected columns in tree grid.

In this demo, the columns with field name `taskName` and `startDate` is frozen using the `isFrozen` property.

### INDEX.TS

```

import { TreeGrid, Freeze } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Freeze);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,

```

```

 height: 317,
 childMapping: 'subtasks',
 allowSelection: false,
 columns: [
 { field: 'taskID', headerText: 'Task ID', textAlign:
'Right', width: 90 },
 { field: 'taskName', headerText: 'Task Name', width: 230,
isFrozen: true },
 { field: 'startDate', headerText: 'Start Date', isFrozen:
true, width: 120, textAlign: 'Right',
 type: 'date', format: { type: 'dateTime', format:
'dd/MM/yyyy' } },
 { field: 'endDate', headerText: 'End Date', width: 150,
textAlign: 'Right',
 type: 'date', format: { type: 'dateTime', format:
'dd/MM/yyyy' } },
 { field: 'duration', headerText: 'Duration', textAlign:
'Right', width: 110 },
 { field: 'progress', headerText: 'Progress', textAlign:
'Right', width: 120 },
 { field: 'priority', headerText: 'Priority', textAlign:
'Left', width: 120 },
 { field: 'approved', headerText: 'Approved', width: 110,
textAlign: 'Left' }
]
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Freeze direction

You can freeze the tree grid columns on the left or right side by using the [column.freeze](#) property and the remaining columns will be movable. The tree grid will automatically move the columns to the left or right position based on the [column.freeze](#) value.

Types of the [column.freeze](#) directions:

- **Left:** Allows you to freeze the columns at the left.
- **Right:** Allows you to freeze the columns at the right.

In this demo, the **Task Name** column is frozen at the left and the **Priority** column is frozen at the right side of the content table.

### INDEX.TS

```

import { TreeGrid, Freeze } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Freeze);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 height: 317,
 treeColumnIndex: 1,
 childMapping: 'subtasks',
 allowSelection: false,
 columns: [

```

```

 { field: 'taskID', headerText: 'Task ID', textAlign:
'Right', width: 90 },
 { field: 'taskName', headerText: 'Task Name', width: 230,
freeze: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 120,
textAlign: 'Right',
 type: 'date', format: { type: 'dateTime', format:
'dd/MM/yyyy' } },
 { field: 'endDate', headerText: 'End Date', width: 150,
textAlign: 'Right',
 type: 'date', format: { type: 'dateTime', format:
'dd/MM/yyyy' } },
 { field: 'duration', headerText: 'Duration', textAlign:
'Right', width: 110 },
 { field: 'progress', headerText: 'Progress', textAlign:
'Right', width: 120 },
 { field: 'priority', headerText: 'Priority', textAlign:
'Left', freeze: 'Right', width: 120 },
 { field: 'approved', headerText: 'Approved', width: 110,
textAlign: 'Left' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* Freeze Direction is not compatible with the [isFrozen](#) and [frozenColumns](#) properties.

#### *Limitations of frozen tree grid*

The following features are not supported in frozen rows and columns:

- Row Template
- Detail Template
- Cell Editing

Freeze Direction feature has the below limitations, along with the above mentioned limitations.

- Infinite scroll cache mode
- Freeze direction in the stacked header is not compatible with column reordering.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript tree grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Virtual scroll in EJ2 JavaScript Treegrid control

TreeGrid allows you to load large amount of data without performance degradation.

To use virtualization, you need to inject `VirtualScroll` module in treegrid.

#### Row virtualization

Row virtualization allows you to load and render rows only in the content viewport. It is an alternative way of paging in which the rows will be appended while scrolling vertically. To setup the row virtualization, you need to define

[enableVirtualization](#) as true and content height by [height](#) property.

The number of records displayed in the TreeGrid is determined implicitly by height of the content area and a buffer records will be maintained in the TreeGrid content in addition to the original set of rows.

Expand and Collapse state of any child record will be persisted.

### INDEX.TS

```
import { VirtualScroll, TreeGrid } from '@syncfusion/ej2-treegrid';
import { virtualData, dataSource } from './datasource.ts';
TreeGrid.Inject(VirtualScroll);
dataSource();
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: virtualData,
 enableVirtualization: true,
 treeColumnIndex: 1,
 childMapping: 'Crew',
 height: 317,
 columns: [
 { field: 'TaskID', headerText: 'Player Jersey', width: 140,
 textAlign: 'Right' },
 { field: 'FIELD1', headerText: 'Player Name', width: 140 },
 { field: 'FIELD2', headerText: 'Year', width: 120,
 textAlign: 'Right' },
 { field: 'FIELD3', headerText: 'Stint', width: 120,
 textAlign: 'Right' },
 { field: 'FIELD4', headerText: 'TMID', width: 120,
 textAlign: 'Right' }
]
 });
treegrid.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Column virtualization

Column virtualization allows you to virtualize columns. It will render column only in the current view port and all other columns are rendered on demand during horizontal scrolling.

To setup the column virtualization, set the

[enableVirtualization](#) and

[enableColumnVirtualization](#) properties as `true`.

### INDEX.TS

```

import { VirtualScroll, TreeGrid } from '@syncfusion/ej2-treegrid';
import { virtualData, dataSource } from './datasource.ts';
TreeGrid.Inject(VirtualScroll);
dataSource();
let treegrid: TreeGrid = new TreeGrid({
 dataSource: virtualData,
 enableVirtualization: true,
 enableColumnVirtualization: true,
 treeColumnIndex: 1,
 childMapping: 'Crew',

```

```

height: 317,
columns: [
 {
 field: 'TaskID',
 headerText: 'Player Jersey',
 width: 140,
 textAlign: 'Right'
 },
 { field: 'FIELD1', headerText: 'Player Name', width: 140 },
 { field: 'FIELD2', headerText: 'Year', width: 120, textAlign: 'Right' },
 { field: 'FIELD3', headerText: 'Stint', width: 120, textAlign: 'Right' }
],
{
 { field: 'FIELD4', headerText: 'TMID', width: 120, textAlign: 'Right' },
 { field: 'FIELD5', headerText: 'LGID', width: 120, textAlign: 'Right' },
 { field: 'FIELD6', headerText: 'GP', width: 120, textAlign: 'Right' },
 { field: 'FIELD7', headerText: 'GS', width: 120, textAlign: 'Right' },
 { field: 'FIELD8', headerText: 'Minutes', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD9', headerText: 'Points', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD10', headerText: 'oRebounds', width: 120, textAlign: 'Right' },
 { field: 'FIELD11', headerText: 'dRebounds', width: 120, textAlign: 'Right' },
 { field: 'FIELD12', headerText: 'Rebounds', width: 120, textAlign: 'Right' },
 { field: 'FIELD13', headerText: 'Assists', width: 120, textAlign: 'Right' },
 { field: 'FIELD14', headerText: 'Steals', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD15', headerText: 'Blocks', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD16', headerText: 'Turnovers', width: 120, textAlign: 'Right' },
 { field: 'FIELD17', headerText: 'PF', width: 120, textAlign: 'Right' },
 { field: 'FIELD18', headerText: 'fgAttempted', width: 120, textAlign: 'Right' },
 { field: 'FIELD19', headerText: 'ftAttempted', width: 120, textAlign: 'Right' },
 { field: 'FIELD20', headerText: 'ThreeAttempted', width: 120, textAlign: 'Right' },
 { field: 'FIELD21', headerText: 'ThreeMade', width: 120, textAlign: 'Right' },
 { field: 'FIELD22', headerText: 'PostGP', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD23', headerText: 'ftMade', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD24', headerText: 'fgMade', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD25', headerText: 'ffmade', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD26', headerText: 'PostGS', width: 120, textAlign: 'Right' }
},
{
 { field: 'FIELD27', headerText: 'PostMinutes', width: 120, textAlign: 'Right' },
 { field: 'FIELD28', headerText: 'PostPoints', width: 120, textAlign: 'Right' }
},

```



```

 { field: 'FIELD29', headerText: 'PostoRebounds', width: 120, textAlign:
 'Right' },
 { field: 'FIELD30', headerText: 'PostdRebounds', width: 120, textAlign:
 'Right' }
]
});
treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Column's [width](#) is required for column virtualization. If column's width is not defined then tree grid will consider its value as **200px**.

#### Limitations for virtualization

- Due to the element height limitation in browsers, the maximum number of records loaded by the treegrid is limited by the browser capability.
- Cell selection will not be persisted in row.
- The page size provided must be two times larger than the number of visible rows in the grid. If the page size is failed to meet this condition then the size will be determined by TreeGrid.
- The virtual height of the treegrid content is calculated using the row height and total number of records in the data source and hence features which changes row height such as text wrapping are not supported. If you want to increase the row height to accommodate the content then you can specify the row height as below to ensure all the table rows are in same height.

```

.e-treegrid .e-row {
height: 2em;
}

```

- Programmatic selection using the **selectRows** method is not supported in virtual scrolling.
- Virtual scrolling is not compatible with Batch editing, clipboard functionality and detail template.
- When virtualization is active in a tree grid, the editCell method is unusable for records outside the currently visible viewport.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Infinite scroll in EJ2 JavaScript Treegrid control

Infinite scrolling is used to load a huge amount of data without degrading the Tree Grid performance. This feature works like the lazy loading concept, which means the buffer data is loaded only when the scrollbar reaches the end of the scroller.

To use Infinite scrolling, set **enableInfiniteScrolling** property as true and inject the **InfiniteScroll** module in the treegrid.

\* In this feature, Tree Grid will not make a new data request when you visit the same page again.

**INDEX.TS**

```
import { InfiniteScroll, TreeGrid } from '@syncfusion/ej2-treegrid';
import { virtualData, dataSource } from './datasource.ts';
TreeGrid.Inject(InfiniteScroll);
dataSource();
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: virtualData,
 enableInfiniteScrolling: true,
 treeColumnIndex: 1,
 childMapping: 'Crew',
 pageSettings: { pageSize: 30 },
 height: 317,
 columns: [
 { field: 'TaskID', headerText: 'Player Jersey', width: 140,
textAlign: 'Right' },
 { field: 'FIELD1', headerText: 'Player Name', width: 140 },
 { field: 'FIELD2', headerText: 'Year', width: 120,
textAlign: 'Right' },
 { field: 'FIELD3', headerText: 'Stint', width: 120,
textAlign: 'Right' },
 { field: 'FIELD4', headerText: 'TMID', width: 120,
textAlign: 'Right' }
]
 });
treegrid.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### InitialBlocks

You can define the initial loading pages count by using `infiniteScrollSettings.initialBlocks` property. By default, this feature loads three pages in initial rendering.

In the below demo, we have changed this property value to load five page records instead of three.

### INDEX.TS

```

import { InfiniteScroll, TreeGrid } from '@syncfusion/ej2-treegrid';
import { virtualData, dataSource } from './datasource.ts';
TreeGrid.Inject(InfiniteScroll);
dataSource();
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: virtualData,
 enableInfiniteScrolling: true,
 infiniteScrollSettings: { initialBlocks: 5 },
 treeColumnIndex: 1,
 childMapping: 'Crew',
 pageSettings: { pageSize: 30 },
 height: 317,
 columns: [
 { field: 'TaskID', headerText: 'Player Jersey', width: 140,
textAlign: 'Right' },
 { field: 'FIELD1', headerText: 'Player Name', width: 140 },
 { field: 'FIELD2', headerText: 'Year', width: 120,
textAlign: 'Right' },

```

```

 { field: 'FIELD3', headerText: 'Stint', width: 120,
textAlign: 'Right' },
 { field: 'FIELD4', headerText: 'TMID', width: 120,
textAlign: 'Right' }
]
 });
 treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Cache Mode

Cache is used to store the loaded rows object in the Tree Grid instance which can be reused for creating the row elements whenever you scroll to already visited page. Also, this mode maintains row elements based on the `infiniteScrollSettings.maxBlocks` count value, once this limit exceeds then it will remove row elements from DOM for new rows.

To enable the cache mode in Infinite scrolling, set `infiniteScrollSettings.enableCache` property as true.

### INDEX.TS

```
import { InfiniteScroll, TreeGrid } from '@syncfusion/ej2-treegrid';
import { virtualData, dataSource } from './datasource.ts';
TreeGrid.Inject(InfiniteScroll);
dataSource();
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: virtualData,
 enableInfiniteScrolling: true,
 infiniteScrollSettings: { enableCache: true },
 treeColumnIndex: 1,
 childMapping: 'Crew',
 pageSettings: { pageSize: 30 },
 height: 317,
 columns: [
 { field: 'TaskID', headerText: 'Player Jersey', width: 140,
 textAlign: 'Right' },
 { field: 'FIELD1', headerText: 'Player Name', width: 140 },
 { field: 'FIELD2', headerText: 'Year', width: 120,
 textAlign: 'Right' },
 { field: 'FIELD3', headerText: 'Stint', width: 120,
 textAlign: 'Right' },
 { field: 'FIELD4', headerText: 'TMID', width: 120,
 textAlign: 'Right' }
]
 });
treegrid.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 TreeGrid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript TreeGrid Control">
 <meta name="author" content="Syncfusion">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Limitations for Infinite Scrolling

- Due to the element height limitation in browsers, the maximum number of records loaded by the tree grid is limited due to the browser capability.
- Initial loading rows total height must be greater than the viewport height.
- Cell selection will not be persisted in cache mode.

- Infinite scrolling is not compatible with batch editing, cell editing, detail template and hierarchy features.
- The aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the
- Programmatic selection using the [selectRows](#) and [selectRow](#) method is not supported in infinite scrolling.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

## Aggregates

### Aggregates in EJ2 JavaScript Treegrid control

Aggregate values are displayed in the TreeGrid footer and in parent row footer for child row aggregate values. It can be configured through `aggregates` property.

[field](#) and [type](#) are the minimum properties required to represent an aggregate column.

To use the aggregate feature, you have to inject the `Aggregate` module.

By default, the aggregate value can be displayed in the treegrid footer, and footer of child rows. To show the aggregate value in one of the cells, use the [footerTemplate](#).

### Built-in aggregate types

The aggregate type should be specified in the [type](#) property to configure an aggregate column.

The built-in aggregates are,

- Sum
- Average
- Min
- Max
- Count
- Truecount
- Falsecount

\* Multiple aggregates can be used for an aggregate column by setting the [type](#) property with an array of aggregate types.

\* Multiple types for a column is supported only when one of the aggregate templates is used.

### Child aggregate

Aggregate value is calculated for child rows, and it is displayed in the parent row footer. Use the [childSummary](#) property to render the child rows aggregate value.

### INDEX.TS

```
import { TreeGrid, Aggregate } from '@syncfusion/ej2-treegrid';
import { summaryRowData } from './datasource.ts';
TreeGrid.Inject(Aggregate);
let treeGridObj: TreeGrid = new TreeGrid(
 {
 dataSource: summaryRowData,
```



```

 childMapping: 'children',
 treeColumnIndex: 0,
 height: 260,
 columns: [
 { field: 'FreightID', headerText: 'Freight ID', width: 130 },
 { field: 'FreightName', width: 195, headerText: 'Freight Name'
 },
 { field: 'UnitWeight', headerText: 'Weight Per Unit', type:
'number', width: 130, textAlign: 'Right' },
 { field: 'TotalUnits', headerText: 'Total Units', type:
'number', width: 125, textAlign: 'Right' }
],
 aggregates: [{
 showChildSummary: true,
 columns: [
 {
 type: 'Max',
 field: 'UnitWeight',
 columnName: 'UnitWeight',
 footerTemplate: 'Maximum: ${Max}'
 },
 {
 type: 'Min',
 field: 'TotalUnits',
 columnName: 'TotalUnits',
 footerTemplate: 'Minimum: ${Min}'
 }
]
 }]
 });
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Footer aggregate in EJ2 JavaScript Treegrid control

Footer aggregate value is calculated for all the rows, and it is displayed in the footer cells. Use the [footerTemplate](#) property to render the aggregate value in footer cells.

#### INDEX.TS

```

import { TreeGrid, Aggregate } from '@syncfusion/ej2-treegrid';
import { summaryRowData } from './datasource.ts';
TreeGrid.Inject(Aggregate);
let treeGridObj: TreeGrid = new TreeGrid(
 {
 dataSource: summaryRowData,
 childMapping: 'children',
 treeColumnIndex: 0,
 height: 260,
 columns: [
 { field: 'FreightID', headerText: 'Freight ID', width: 130 },
 { field: 'FreightName', width: 195, headerText: 'Freight Name' }
]
 },

```

```

 { field: 'UnitWeight', headerText: 'Weight Per Unit', type:
'number', width: 130, textAlign: 'Right' },
 { field: 'TotalUnits', headerText: 'Total Units', type:
'number', width: 125, textAlign: 'Right' }
],
 aggregates: [{
 showChildSummary: false,
 columns: [
 {
 type: 'Max',
 field: 'UnitWeight',
 columnName: 'UnitWeight',
 footerTemplate: 'Maximum: ${Max}'
 },
 {
 type: 'Min',
 field: 'TotalUnits',
 columnName: 'TotalUnits',
 footerTemplate: 'Minimum: ${Min}'
 }
]
 }]
 });
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The aggregate values must be accessed inside the template using their corresponding [type](#) name.

#### *How to format aggregate value*

You can format the aggregate value result by using the [format](#) property.

#### **INDEX.TS**

```

import { TreeGrid, Aggregate } from '@syncfusion/ej2-treegrid';
import { summaryData } from './datasource.ts';
TreeGrid.Inject(Aggregate);
let treeGridObj: TreeGrid = new TreeGrid(
 {
 dataSource: summaryData,
 childMapping: 'subtasks',
 treeColumnIndex: 0,
 height: 260,
 columns: [
 { field: 'category', headerText: 'Category', width: 160 },
 { field: 'units', headerText: 'Total Units', width: 130, type:
'number', textAlign: 'Right' },
 { field: 'unitPrice', headerText: 'Unit Price($)', width: 110,
type: 'number', format: 'C2', textAlign: 'Right' },
 { field: 'price', headerText: 'Price($)', width: 160, textAlign:
'Right', type: 'number', format: 'C2' },
],
 aggregates: [{
 columns: [
 {
 type: 'Sum',
 field: 'price',

```

```

 columnName: 'price',
 format: 'C2',
 footerTemplate: 'Total: ${Sum}'
 }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>

```

```

 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Custom aggregate in EJ2 JavaScript Treegrid control

To calculate the aggregate value with your own aggregate functions, use the custom aggregate option.

To use custom aggregation, specify the [type](#) as `Custom`, and provide the custom aggregate function in the [customAggregate](#) property.

#### INDEX.TS

```

import { TreeGrid, Aggregate } from '@syncfusion/ej2-treegrid';
import { summaryData } from './datasource.ts';
import { getObject, CustomSummaryType } from '@syncfusion/ej2-grids';
let customAggregateFn: CustomSummaryType = (data: Object): number => {
 let sampleData: Object[] = getObject('result', data);
 let countLength: number; countLength = 0;
 sampleData.filter((item: Object) => {
 let data: string = getObject('category', item);
 if (data === 'Frozen seafood') {
 countLength++;
 }
 }));
 return countLength;
};
TreeGrid.Inject(Aggregate);
let treeGridObj: TreeGrid = new TreeGrid(
 {
 dataSource: summaryData,
 childMapping: 'subtasks',
 width: 'auto',
 height: 245,
 treeColumnIndex: 1,
 columns: [
 { field: 'category', headerText: 'Category', width: 200 },
 { field: 'units', headerText: 'Total Units', width: 130, type:
'number', textAlign: 'Right' },
 { field: 'unitPrice', headerText: 'Unit Price($)', width: 110,
type: 'number', format: 'C2', textAlign: 'Right' },
 { field: 'price', headerText: 'Price($)', width: 110, textAlign:
'Right', type: 'number', format: 'C' },
],
 aggregates: [{
 showChildSummary: false,
 columns: [{
 type: 'Custom',
 customAggregate: customAggregateFn,
 columnName: 'category',
 footerTemplate: 'Count of Frozen seafood : ${Custom}'
 }]
 }]
 }
);

```

```

 }}
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

 ele.style.visibility = "visible";
 }
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To access the custom aggregate value inside the template, use the key as **Custom**.

### Print in EJ2 JavaScript Treegrid control

To print the TreeGrid, use the [print](#) method from treegrid instance. The print option can be displayed on the [toolbar](#) by adding the **print** toolbar item.

#### INDEX.TS

```

import { TreeGrid, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Print'],
 height: 265,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Page setup

Some of the print options cannot be configured through JavaScript code. So, you have to customize the layout, paper size, and margin options using the browser page setup dialog. Please refer to the following links to know more about the browser page setup:

- [Chrome](#)
- [Firefox](#)
- [Safari](#)
- [IE](#)

### Print using an external button

To print the treegrid from an external button, invoke the [print](#) method.

**INDEX.TS**

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { Button } from '@syncfusion/ej2-buttons';
import { sampleData } from './datasource.ts';
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 height: 265,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
let printBtn: Button = new Button();
printBtn.appendTo('#print');
document.getElementById('print').addEventListener('click', () => {
 treeGridObj.print();
});
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="print">Print</button>
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Print the visible page

By default, the treegrid prints all the pages. To print the current page alone, set the [printMode](#) to **CurrentPage**.

### INDEX.TS

```

import { TreeGrid, Toolbar, Page } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Print'],
 printMode: 'CurrentPage',
 allowPaging: true,
 pageSettings: { pageSize: 8 },
 height: 220,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },

```

```

 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```

<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Print large number of columns

By default, the browser uses A4 as page size option to print pages and to adapt the size of the page the browser print preview will auto-hide the overflowed contents. Hence treegrid with large number of columns will cut off to adapt the print page.

To show large number of columns when printing, adjust the scale option from print option panel based on your content size.



### Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the treegrid using [toolbarClick](#) and [printComplete](#) events.

In the `toolbarClick` event, based on `args.item.text` as `Print`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `printComplete` event, We have reversed the state back to the previous state.

In the below example, we have `Duration` as a hidden column in the treegrid. While printing, we have changed `Duration` to visible column and `StartDate` as hidden column.

### INDEX.TS

```

import { TreeGrid, Toolbar } from '@syncfusion/ej2-treegrid';
import { Column } from '@syncfusion/ej2-grids';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar);

```

```

let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Print'],
 toolbarClick: function(args) {
 if (args.item.text === 'Print') {
 let cols: Column[] = this.grid.columns;
 for (var i = 0; i < cols.length; i++) {
 if (cols[i].field === "duration") {
 cols[i].visible = true;
 }
 else if (cols[i].field === "startDate") {
 cols[i].visible = false;
 }
 }
 }
 },
 printComplete: function(args) {
 let cols: Column[] = this.grid.columns;
 for (var i = 0; i < cols.length; i++) {
 if (cols[i].field === "duration") {
 cols[i].visible = false;
 }
 else if (cols[i].field === "StartDate") {
 cols[i].visible = true;
 }
 }
 }
 height: 265,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', visible: false,
width: 80, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Limitations of Printing Large Data

When treegrid contains large number of data, printing all the data at once is not a best option for the browser performance. Because to render all the DOM elements in one page will produce performance issues in the browser. It leads to browser slow down or browser hang.

If printing of all the data is still needed, we suggest to Export the treegrid to **Excel** or **CSV** or **Pdf** and then print it from another non-web based application.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

## State Persistence

### State persistence in EJ2 JavaScript Treegrid control

State persistence refers to the TreeGrid's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser.

State persistence stores treegrid's model object in the local storage when the [enablePersistence](#) is defined as true.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Get or set local storage value in EJ2 JavaScript Treegrid control

If the [enablePersistence](#) property is set to true, the treegrid property value is saved in the window.localStorage for reference. You can get/set the localStorage value by using the `getItem/setItem` method in the `window.localStorage`.

```
`ts
```

```
//get the TreeGrid model.
```

```
let value: string = window.localStorage.getItem('treegridTreeGrid'); // "treegridTreeGrid" is component name + component id.
```

```
let model: Object = JSON.parse(model);
```

```
`
```

```
`ts
```

```
//set the TreeGrid model.
```

```
window.localStorage.setItem('treegridTreeGrid', JSON.stringify(model)); // "treegridTreeGrid" is component name + component id.
```

```
`
```

## Tool Bar

### Tool bar in EJ2 JavaScript Treegrid control

The TreeGrid provides ToolBar support to handle treegrid actions. The [toolbar](#) property accepts either the collection of built-in toolbar items and [ItemModel](#) objects for custom toolbar items or HTML element ID for toolbar template.

To use ToolBar, inject `Toolbar` module in the treegrid.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.



## Tool bar items in EJ2 JavaScript Treegrid control

### Built-in toolbar items

Built-in toolbar items execute standard actions of the treegrid, and it can be added by defining the [toolbar](#) as a collection of built-in items. It renders the button with icon and text.

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items   Actions	
----- -----	
ExpandAll	Expands all the rows.
CollapseAll	Collapses all the rows.
Add	Adds a new record.
Edit	Edits the selected record.
Update	Updates the edited record.
Delete	Deletes the selected record.
Cancel	Cancel the edit state.
Search	Searches the records by the given key.
Print	Prints the treegrid.
ExcelExport	Exports the treegrid to Excel.
PdfExport	Exports the treegrid to PDF.
WordExport	Exports the treegrid to Word.
Indent	Indents the record to one level of hierarchy.
Outdent	Outdents the record to one level of hierarchy.

### INDEX.TS

```
import { TreeGrid, Toolbar, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['Print', 'Search'],
 height: 265,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
```

```

]
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {

```

```

 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* The [toolbar](#) has options to define both built-in and custom toolbar items.

#### *Enable or disable toolbar items*

You can enable/disable toolbar items by using the `enableItems` method.

#### **INDEX.TS**

```

import { TreeGrid, Toolbar, Filter } from '@syncfusion/ej2-treegrid';
import { Button } from '@syncfusion/ej2-buttons';
import { sampleData } from './datasource.ts';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
TreeGrid.Inject(Toolbar, Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowFiltering: true,
 toolbar: ['QuickFilter', 'ClearFilter'],
 toolbarClick: clickHandler,
 height: 200,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
let enable: Button = new Button({}, '#enable');
let disable: Button = new Button({}, '#disable');
enable.element.onclick = () => {
 treeGridObj.toolbarModule.enableItems([treeGridObj.element.id +
'_gridcontrol_QuickFilter', treeGridObj.element.id +
'_gridcontrol_ClearFilter'], true); // enable toolbar items.
};
disable.element.onclick = () => {
 treeGridObj.toolbarModule.enableItems([treeGridObj.element.id +
'_gridcontrol_QuickFilter', treeGridObj.element.id +
'_gridcontrol_ClearFilter'], false); // disable toolbar items.
};
function clickHandler(args: ClickEventArgs): void {
 if (args.item.text === 'QuickFilter') {
 treeGridObj.filterByColumn('taskName', 'startswith', 'Testing');
 }
}

```

```

 if (args.item.text === 'ClearFilter') {
 treeGridObj.clearFiltering();
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="enable" class="e-flat">Enable</button>
 <button id="disable" class="e-flat">Disable</button>
 <div id="TreeGrid"></div>
 </div>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Custom tool bar in EJ2 JavaScript Treegrid control

Custom toolbar items can be added by defining the [toolbar](#) as a collection of [ItemModels](#).

Actions for this customized toolbar items are defined in the [toolbarClick](#) event.

By default, Custom toolbar items are in position **Left**. You can change the position by using the [align](#) property. In the below sample, we have applied position **Right** for the **Quick Filter** toolbar item.

#### INDEX.TS

```
import { TreeGrid, Toolbar, Filter } from '@syncfusion/ej2-treegrid';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: [{text: 'Quick Filter', tooltipText: 'Quick Filter', id:
'toolbarfilter', align:'Right'}],
 toolbarClick: (args: ClickEventArgs) => {
 if (args.item.id === 'toolbarfilter') {
 treeGridObj.filterByColumn('taskName', 'startswith', 'Testing');
 }
 },
 allowFiltering: true,
 height: 220,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* The [toolbar](#) has options to define both built-in and custom toolbar items.

\* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

*Built-in and custom items in toolbar*

TreeGrid have an option to use both built-in and custom toolbar items at same time.

In the below example, **ExpandAll**, **CollapseAll** are built-in toolbar items and **Click** is custom toolbar item.

**INDEX.TS**

```
import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { EmitType } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let clickHandler: EmitType<ClickEventArgs> = (args: ClickEventArgs) => {
 if (args.item.text === 'Click') {
 alert("Custom toolbar click...");
 }
};
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 toolbar: ['ExpandAll', 'CollapseAll', { text: 'Click', tooltipText:
'Click', prefixIcon: 'e-time', id: 'Click' }],
 toolbarClick: clickHandler,
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting:
true },
 height: 270,
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Pdf Export

### Pdf export in EJ2 JavaScript Treegrid control

PDF export allows exporting TreeGrid data to PDF document. You need to use the

[pdfExport](#) method for exporting. To enable PDF export in the treegrid, set the [allowPdfExport](#) as true.

To use PDF export, inject the PdfExport module in treegrid.

### INDEX.TS



```

import { TreeGrid, Page, Toolbar, PdfExport } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90, textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 treeGridObj.pdfExport();
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### <!-- Multiple exporting

PDF export provides an option for exporting multiple treegrids to same file. In this exported document, each treegrid will be exported to new page of document in same file.

### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport } from '@syncfusion/ej2-
treegrid';
import { sampleData, projectData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let firstTreeGrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 allowPdfExport: true,
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [

```

```

 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
firstTreeGrid.appendTo('#TreeGrid');
let secondTreeGrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: 'TaskID',
 parentIdMapping: 'parentID',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 treeColumnIndex: 1,
 pageSettings: {pageSize: 7},
 columns: [
 { field: 'TaskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 {
 field: 'StartDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date',format: 'yMd'
 },
 { field: 'Duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
secondTreeGrid.appendTo('#TreeGrid2');
firstTreeGrid.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let firstGridPdfExport: Promise<Object> = firstTreeGrid.pdfExport({},
true);
 firstGridPdfExport.then((pdfData: Object) => {
 secondTreeGrid.pdfExport({}, false, pdfData);
 });
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <p>First Grid</p>
 <div id="TreeGrid"></div>
 <p>Second Grid</p>
 <div id="TreeGrid2"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

--&gt;

&lt;!-- Export current page

PDF export provides an option to export the current page into PDF. To export current page, define the `exportType` to `CurrentPage`.

### INDEX.TS

```
import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 exportType: 'CurrentPage'
 };
 treeGridObj.pdfExport(exportProperties);
 }
}
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

-->

#### Custom data source

PDF export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`

#### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({

```

```

 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
 });
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 dataSource: sampleData,
 };
 treeGridObj.pdfExport(exportProperties);
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Pdf export options in EJ2 JavaScript Treegrid control

#### *Export hidden columns*

PDF export provides an option to export hidden columns of TreeGrid by defining the `includeHiddenColumn` as `true`.

#### **INDEX.TS**

```

import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },

```



```

 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', visible: false,
width: 80, textAlign: 'Right' }
]
 });
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 includeHiddenColumn: true
 };
 treeGridObj.pdfExport(exportProperties);
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

#### Show or hide columns on exported PDF

You can show a hidden column or hide a visible column while exporting the treegrid using [toolbarClick](#) and [pdfExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.text` as `PDF Export`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `pdfExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have `Duration` as a hidden column in the treegrid. While exporting, we have changed `Duration` to visible column and `StartDate` as hidden column.

#### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-treegrid';
import { Column } from '@syncfusion/ej2-grids';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [

```

```

 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', visible: false,
width: 80, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
if (args['item'].text === 'PDF Export') {
 let cols: Column[] = treeGridObj.grid.columns;
 cols[2].visible = false;
 cols[3].visible = true;
 treeGridObj.pdfExport();
}
}
treeGridObj.pdfExportComplete = () => {
 let cols: Column[] = treeGridObj.grid.columns;
 cols[3].visible = false;
 cols[2].visible = true;
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### How to change page orientation

Page orientation can be changed Landscape(Default Portrait) for the exported document using the `exportProperties`.

### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'

```

```

 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 pageOrientation: 'Landscape',
 };
 treeGridObj.pdfExport(exportProperties);
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>

```

```
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

#### *How to change page size*

Page size can be customized for the exported document using the `exportProperties`.

Supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- Archa
- Archb
- Archc
- Archd
- Arche
- Flsa
- HalfLetter
- Letter11x17
- Ledger

**INDEX.TS**

```
import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 pageSize: 'Letter'
 };
 treeGridObj.pdfExport(exportProperties);
 }
}
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### *To customize PDF export*

PDF export provides an option to customize mapping of treegrid to exported PDF document.

### *File name for exported document*

You can assign the file name for the exported document by defining `fileName` property in `PdfExportProperties`.

### **INDEX.TS**

```

import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,

```



```

 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
 });
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 fileName: "new.xlsx"
 };
 treeGridObj.pdfExport(exportProperties);
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Font customization

#### Default fonts for PDF exporting

By default, treegrid uses Helvetica font in the exported document. You can change the default font by using `pdfExportProperties.theme` property. The available default fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

The code example for changing default font,

```
`ts
```

```
import { PdfStandardFont, PdfFontFamily, PdfFontStyle } from '@syncfusion/ej2-pdf-export';
```

```
...
```

```
let pdfExportProperties: PdfExportProperties = {
```

```
theme: {
```

```

header: {font: new PdfStandardFont(PdfFontFamily.TimesRoman, 11, PdfFontStyle.Bold),
record: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 10) }
}
};
`

```

#### Add custom font for PDF exporting

You can change the default font of TreeGrid header, content and caption cells in the exported document by using `pdfExportProperties.theme` property.

In the following example, we have used Advent Pro font to export the treegrid with Hungarian fonts.

#### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
'@syncfusion/ej2-treegrid';
import { sampleData, adventProFont } from './datasource.ts';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 theme: {
 header: {font: new PdfTrueTypeFont(adventProFont, 12) },
 record: { font: new PdfTrueTypeFont(adventProFont, 9) }
 }
 };
 treeGridObj.pdfExport(exportProperties);
 }
}
}

```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

**PdfTrueTypeFont** accepts base 64 format of the Custom Font.

Pdf cell style customization in EJ2 JavaScript Treegrid control

#### *Conditional cell formatting*

TreeGrid cells in the exported PDF can be customized or formatted using [pdfQueryCellInfo](#) event. In this event, we can format the treegrid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for **Duration** column in the exported document by **args.cell** and **backgroundColor** property.

#### **INDEX.TS**

```
import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties,
RowDataBoundEventArgs, PdfQueryCellInfoEventArgs } from '@syncfusion/ej2-
treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 treeGridObj.pdfExport();
 }
}
treeGridObj.pdfQueryCellInfo = (args: PdfQueryCellInfoEventArgs) => {
 if (args.column.field === 'duration') {
 if (+args.value === 0 || args.value === '') {
 args.style = {backgroundColor: '#3366cc'};
 }
 else if (args.value < 3) {
 args.style = {backgroundColor: '#7b2b1d'};
 }
 }
}
```

```
treeGridObj.queryCellInfo = (args: RowDataBoundEventArgs) => {
 if (args.data['duration'] == 0 && args.column.field === 'duration') {
 args.cell.style.background= '#336c12';
 } else if (args.data['duration'] < 3 && args.column.field ===
'duration') {
 args.cell.style.background= '#7b2b1d';
 }
}
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
```

```

 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Theme

PDF export provides an option to include theme for exported PDF document.

To apply theme in exported PDF, define the **theme** in **exportProperties**.

### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExportProperties, PdfExport } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['PdfExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let exportProperties: PdfExportProperties = {
 theme: {
 header: {
 fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true, border: { color: '#64FA50', lineStyle: 'Thin' }
 },
 record: {
 fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true

```

```

 }
 }
};
treeGridObj.pdfExport (exportProperties);
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>

```



```

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

By default, material theme is applied to exported PDF document.

### Adding header and footer in EJ2 JavaScript Treegrid control

You can customize text, page number, line, page size and changing orientation in header and footer.

#### *How to write a text in header or footer*

You can add text either in Header or Footer of exported PDF document.

```
`ts
```

```

let exportProperties: PdfExportProperties = {
 header: {
 fromTop: 0,
 height: 130,
 contents: [
 {
 type: 'Text',
 value: "Task Details",
 position: { x: 0, y: 50 },
 style: { textBrushColor: '#000000', fontSize: 13 }
 },
]
 }
},
`

```

#### *How to draw a line in header or footer*

you can add line either in Header or Footer of the exported PDF document.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

```
`ts
```

```
let exportProperties: PdfExportProperties = {
 header: {
 fromTop: 0,
 height: 130,
 contents: [
 {
 type: 'Line',
 style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
 points: { x1: 0, y1: 4, x2: 685, y2: 4 }
 }
]
 }
},
```

#### *Add page number in header or footer*

you can add page number either in Header or Footer of exported PDF document.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

`ts

```
let exportProperties: PdfExportProperties = {
 header: {
 fromTop: 0,
 height: 130,
 contents: [
 {
 type: 'PageNumber',
 pageNumberType: 'Arabic',
 format: 'Page { $current } of { $total }', //optional
 position: { x: 0, y: 25 },
 style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
```

```

}
]
}
}
`

```

#### *Insert an image in header or footer*

Image (Base64 string) can be added in the exported document in header/footer using the `exportProperties`.

```

`ts
let exportProperties: PdfExportProperties = {
 header: {
 fromTop: 0,
 height: 130,
 contents: [
 {
 type: 'Image',
 src: image,
 position: { x: 40, y: 10 },
 size: { height: 100, width: 250 },
 }
]
 }
}
`

```

The below code illustrates the pdf export customization.

#### **INDEX.TS**

```

import { TreeGrid, Page, Toolbar, PdfExport, PdfExportProperties } from
 '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
import { image } from './image.ts';
TreeGrid.Inject(Page, Toolbar, PdfExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPdfExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['PdfExport'],

```

```

 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
 });
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'PDF Export') {
 let pdfExportProperties: PdfExportProperties = {
 header: {
 fromTop: 0,
 height: 130,
 contents: [
 {
 type: 'Image',
 src: image,
 position: { x: 40, y: 10 },
 size: { height: 100, width: 250 },
 }
]
 },
 footer: {
 fromBottom: 160,
 height: 150,
 contents: [
 {
 type: 'PageNumber',
 pageNumberType: 'Arabic',
 format: 'Page {$current} of {$total}',
 position: { x: 0, y: 25 },
 style: { textBrushColor: '#ffff80', fontSize: 15 }
 }
]
 }
 };
 treeGridObj.pdfExport(pdfExportProperties);
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">

```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Exporting Tree Grid in Server

The Tree Grid have an option to export the data to PDF in server side using tree grid server export library.

### *Server dependencies*

The Server side export functionality is shipped in the Syncfusion.EJ2.TreeGridExport package, which is available in Essential Studio and [nuget.org](https://www.nuget.org). The following list of dependencies is required for tree grid server side PDF exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.TreeGridExport

### *Server configuration*

The following code snippet shows server configuration using ASP.NET MVC Controller Action.

To Export the tree grid in server side, You need to call the [serverPdfExport](#) method for passing the tree grid properties to server exporting action.

```
`ts
public IActionResult ServerSideExporting()
{
 var order = TreeData.GetDefaultData();
 ViewBag.dataSource = order;
 return View();
}

public IActionResult PdfExport(string treeGridModel)
{
 if (treeGridModel == null)
 {
 return View();
 }

 TreeGridExcelExport exp = new TreeGridExcelExport();
 Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
 return exp.ExportToPdf<TreeData>(gridProperty, TreeData.GetDefaultData());
}

private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
 Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
 (Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
 typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));

 TreeGridColumnModel cols =
 (TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
 typeof(TreeGridColumnModel));
}
```

```
TreeGridModel.Columns = cols.columns;
return TreeGridModel;
}
public class TreeGridColumnModel
{
public List<TreeGridColumn> columns { get; set; }
}
`
`ts
import { TreeGrid, Toolbar } from '@syncfusion/ej2-treegrid';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
TreeGrid.Inject(Toolbar);
let data: DataManager = new DataManager({
url: "Home/DataSource",
adaptor: new UrlAdaptor
});
let treegrid: TreeGrid = new TreeGrid({
dataSource: data,
parentIdMapping: 'ParentItem',
hasChildMapping: 'isParent',
toolbar: ['PdfExport'],
columns: [
{ field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width: 90 },
{ field: 'TaskName', headerText: 'Task Name', width: 180 },
{ field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 90, format: { skeleton: 'yMd', type: 'date' } },
{ field: 'Duration', headerText: 'Duration', width: 80, textAlign: 'Right' }
],
height: 265
});
treegrid.appendTo('#TreeGrid');
treegrid.toolbarClick = (args: Object) => {
if (args['item'].id === 'TreeGridgridcontrolpdfexport') {
```

```

treegrid.serverPdfExport("Home/PdfExport");
}
}
`

```

*Rotate a header text to a certain degree in the exported tree grid on the server side*

The Tree Grid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported PDF file. To achieve this requirement, define the `BeginCellLayout` event of the `PdfExportProperties` with an event handler to perform the required action.

The `PdfHeaderCellRendering` will be triggered when creating a column header for the pdf document to be exported. Collect the column header details in this event and handle the custom in the `BeginCellLayout` event handler.

In the following demo, the `DrawString` method from the `Graphics` is used to rotate the header text of the column header inside the `BeginCellLayout` event handler.

**Note:** A PDF exporting is not supported to rotate the column header on the client side.

```

`ts
public IActionResult PdfExport(string treeGridModel)
{
 if (treeGridModel == null)
 {
 return View();
 }

 TreeGridPdfExport exp = new TreeGridPdfExport();
 TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
 gridProperty.PdfHeaderCellRendering = PdfHeaderQueryCellInfo;
 PdfGrid grid = new PdfGrid();

 Syncfusion.EJ2.TreeGridExport.PdfExportProperties pdfExportProperties = new
 Syncfusion.EJ2.TreeGridExport.PdfExportProperties();
 pdfExportProperties.IsRepeatHeader = true;
 exp.BeginCellLayout = new PdfGridBeginCellLayoutEventHandler(BeginCellEvent);
 System.Collections.IEnumerable data = Syncfusion.EJ2.Base.Dat.DataTableToJson(ViewBag.dataSource);
 var result = exp.PdfExport<dynamic>(gridProperty, data, pdfExportProperties);
 return View();
}

public void BeginCellEvent(object sender, PdfGridBeginCellLayoutEventArgs args)
{

```



```

PdfGrid grid = (PdfGrid)sender;
var brush = new PdfSolidBrush(new PdfColor(Color.DimGray));
args.Graphics.Save();
args.Graphics.TranslateTransform(args.Bounds.X + 50, args.Bounds.Height + 40); // give the value for
bounds x and Y by the user
args.Graphics.RotateTransform(-60); // give the rotate degree value by the user
// Draw the text at particular bounds.
args.Graphics.DrawString(headerValues[args.CellIndex], new PdfStandardFont(PdfFontFamily.Helvetica,
10), brush, new PointF(0, 0));
if (args.IsHeaderRow)
{
grid.Headers[0].Cells[args.CellIndex].Value = string.Empty;
}
args.Graphics.Restore();
}
private void PdfHeaderQueryCellInfo(object pdf)
{
Syncfusion.EJ2.TreeGridExport.PdfHeaderCellEventArgs name =
(Syncfusion.EJ2.TreeGridExport.PdfHeaderCellEventArgs)pdf;
PdfGrid grid = new PdfGrid();
headerValues.Add(name.Column.HeaderText);
var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 6);
SizeF size = font.MeasureString(longestString);
name.Headers[0].Height = size.Width * 2;
}
,

```

## Excel Export

### Excel export in EJ2 JavaScript Treegrid control

The excel export allows exporting TreeGrid data to Excel document. You need to use the [excelExport](#) method for exporting. To enable Excel export in the treegrid, set the [allowExcelExport](#) as true.

To use excel export, You need to inject the `ExcelExport` module in treegrid.

### INDEX.TS

```

import { TreeGrid, Toolbar, ExcelExport, Page } from '@syncfusion/ej2-
treegrid';
import { sampleData } from '../datasource.ts';

```

```

TreeGrid.Inject(Toolbar, ExcelExport, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 treeGridObj.excelExport();
 }
}
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Persist collapsed state

You can persist the collapsed state in the exported document by defining `isCollapsedStatePersist` property as true in `TreeGridExcelExportProperties` parameter of [excelExport](#) method.

### INDEX.TS

```

import { TreeGrid, Toolbar, ExcelExport, TreeGridExcelExportProperties, Page
} from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, ExcelExport, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },

```

```

 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 let excelExportProperties: TreeGridExcelExportProperties = {
 isCollapsedStatePersist: true
 };
 treeGridObj.excelExport(excelExportProperties);
 }
}
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Custom data source

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`.

### INDEX.TS

```

import { TreeGrid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, ExcelExport, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {

```

```
 let excelExportProperties: ExcelExportProperties = {
 dataSource: sampleData
 };
 treeGridObj.excelExport(excelExportProperties);
}
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
```

```

 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

Excel export options in EJ2 JavaScript Treegrid control

*To customize excel export*

The excel export provides an option to customize mapping of the treegrid to excel document.

[Export hidden columns](#)

The excel export provides an option to export hidden columns of treegrid by defining `includeHiddenColumn` as `true`.

## INDEX.TS

```

import { TreeGrid, Toolbar, ExcelExport, Page, ExcelExportProperties } from
 '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, ExcelExport, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
 textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', visible: false,
 width: 80, textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 let excelExportProperties: ExcelExportProperties = {
 includeHiddenColumn: true
 };
 }
};

```

```

 treeGridObj.excelExport(excelExportProperties);
 }
}
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');

```



```

if(ele) {
 ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

#### Show or hide columns on exported excel

You can show a hidden column or hide a visible column while printing the treegrid using [toolbarClick](#) and [excelExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.text` as `Excel Export`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `excelExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have `Duration` as a hidden column in the treegrid. While exporting, we have changed `Duration` to visible column and `StartDate` as hidden column.

#### INDEX.TS

```

import { TreeGrid, Page, Toolbar, ExcelExport, ExcelExportProperties } from
 '@syncfusion/ej2-treegrid';
import { Column } from '@syncfusion/ej2-grids';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, ExcelExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
 textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', visible: false,
 width: 80, textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 let cols: Column[] = treeGridObj.grid.columns;
 cols[2].visible = false;
 cols[3].visible = true;
 treeGridObj.excelExport();
 }
}

```

```
treeGridObj.excelExportComplete = () => {
 let cols: Column[] = treeGridObj.grid.columns;
 cols[3].visible = false;
 cols[2].visible = true;
}
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js" type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### File name for exported document

You can assign the file name for the exported document by defining `fileName` property in [ExcelExportProperties](#).

### INDEX.TS

```
import { TreeGrid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
'@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, ExcelExport, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 let excelExportProperties: ExcelExportProperties = {
 fileName: "new.xlsx"
 };
 treeGridObj.excelExport(excelExportProperties);
 }
}
treeGridObj.appendTo('#TreeGrid');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Grid</title>
```

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

Excel cell style customization in EJ2 JavaScript Treegrid control

### Conditional cell formatting

TreeGrid cells in the exported Excel can be customized or formatted using [excelQueryCellInfo](#) event. In this event, we can format the treegrid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for **Duration** column in the exported excel by **args.cell** and **backgroundColor** property.

### INDEX.TS

```
import { TreeGrid, Page, Toolbar, ExcelExport, ExcelExportProperties,
RowDataBoundEventArgs, ExcelQueryCellInfoEventArgs } from '@syncfusion/ej2-
treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, ExcelExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 treeGridObj.excelExport();
 }
}
treeGridObj.excelQueryCellInfo = (args: ExcelQueryCellInfoEventArgs) => {
 if (args.column.field === 'duration') {
 if (args.value === 0 || args.value === "") {
 args.style = { backgroundColor: '#336c12' };
 }
 else if (args.value < 3) {
 args.style = { backgroundColor: '#7b2b1d' };
 }
 }
}
treeGridObj.queryCellInfo = (args: RowDataBoundEventArgs) => {
 if (args.data['duration'] === 0 && args.column.field === 'duration') {
 args.cell.style.backgroundColor= '#336c12';
 }
}
```

```
 } else if (args.data['duration'] < 3 && args.column.field ===
'duration') {
 args.cell.style.background= '#7b2b1d';
 }
}
treeGridObj.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Theme

The excel export provides an option to include theme for exported excel document.

To apply theme in exported Excel, define the **theme** in **exportProperties**.

### INDEX.TS

```
import { TreeGrid, Page, Toolbar, ExcelExportProperties, ExcelExport } from
 '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, ExcelExport);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: {pageSize: 7},
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
 textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 let exportProperties: ExcelExportProperties = {
 theme: {
 header: { fontName: 'Segoe UI', fontColor: '#666666' },
 record: { fontName: 'Segoe UI', fontColor: '#666666' },
 caption: { fontName: 'Segoe UI', fontColor: '#666666' }
 }
 };
 treeGridObj.excelExport(exportProperties);
 }
}
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

By default, material theme is applied to exported excel document.

### Adding header and footer in EJ2 JavaScript Treegrid control

The excel export provides an option to include header and footer content for exported excel document.

#### INDEX.TS

```
import { TreeGrid, Toolbar, ExcelExport, ExcelExportProperties, Page } from
 '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, ExcelExport, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowExcelExport: true,
 allowPaging: true,
 height: 220,
 pageSettings: { pageSize: 7 },
 toolbar: ['ExcelExport'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.toolbarClick = (args: Object) => {
 if (args['item'].text === 'Excel Export') {
 let excelExportProperties: ExcelExportProperties = {
 header: {
 headerRows: 7,
 rows: [
 { cells: [{ colSpan: 4, value: "Northwind Traders",
style: { fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, }
}] },
 { cells: [{ colSpan: 4, value: "2501 Aerial Center
Parkway", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center',
bold: true, } }] },
 { cells: [{ colSpan: 4, value: "Suite 200 Morrisville,
NC 27560 USA", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } }] },
 { cells: [{ colSpan: 4, value: "Tel +1 888.936.8638 Fax
+1 919.573.0306", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } }] },
]
 }
 };
 }
};
```

```

 { cells: [{ colSpan: 4, hyperlink: { target:
'https://www.northwind.com/', displayText: 'www.northwind.com' }, style: {
hAlign: 'Center' } }] },
 { cells: [{ colSpan: 4, hyperlink: { target:
'mailto:support@northwind.com' }, style: { hAlign: 'Center' } }] },
],
 footer: {
 footerRows: 4,
 rows: [
 { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } }] },
 { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } }] }
]
 },
};
treeGridObj.excelExport(excelExportProperties);
}
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Exporting Tree Grid in Server

The Tree Grid have an option to export the data to Excel in server side using tree grid server export library.

#### Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.TreeGridExport package, which is available in Essential Studio and [nuget.org](https://nuget.org). The following list of dependencies is required for tree grid server side Excel exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.TreeGridExport

#### Server configuration

The following code snippet shows server configuration using ASP.NET MVC Controller Action.

To Export the tree grid in server side, You need to call the [serverExcelExport](#) method for passing the tree grid properties to server exporting action.

```

`ts
public IActionResult ServerSideExporting()
{
 var order = TreeData.GetDefaultData();
 ViewBag.dataSource = order;
 return View();
}

```

```

public IActionResult ExcelExport(string treeGridModel)
{
 if (treeGridModel == null)
 {
 return View();
 }
 TreeGridExcelExport exp = new TreeGridExcelExport();
 Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
 return exp.ExportToExcel<TreeData>(gridProperty, TreeData.GetDefaultData());
}

private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
 Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
 (Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
 typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));

 TreeGridColumnModel cols =
 (TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
 typeof(TreeGridColumnModel));

 TreeGridModel.Columns = cols.columns;
 return TreeGridModel;
}

public class TreeGridColumnModel
{
 public List<TreeGridColumn> columns { get; set; }
}
`typescript
import { TreeGrid, Toolbar } from '@syncfusion/ej2-treegrid';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
TreeGrid.Inject(Toolbar);
let data: DataManager = new DataManager({
 url: "Home/DataSource",
 adaptor: new UrlAdaptor
});

```

```

let treegrid: TreeGrid = new TreeGrid({
 dataSource: data,
 parentIdMapping: 'ParentItem',
 hasChildMapping: 'isParent',
 toolbar: ['ExcelExport'],
 columns: [
 { field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width: 90 },
 { field: 'TaskName', headerText: 'Task Name', width: 180 },
 { field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 90, format: { skeleton: 'yMd', type: 'date' } },
 { field: 'Duration', headerText: 'Duration', width: 80, textAlign: 'Right' }
],
 height: 265
});
treegrid.appendTo('#TreeGrid');
treegrid.toolbarClick = (args: Object) => {
 if (args['item'].id === 'TreeGridgridcontrolexcelexport') {
 treegrid.serverExcelExport("Home/ExcelExport");
 }
}

```

#### *CSV Export in server side*

You can export the tree grid to CSV format by using the [serverCsvExport](#) method which will pass the tree grid properties to server.

In the below demo, we have invoked the above method inside the [toolbarClick](#) event. In server side, we have deserialized the tree grid properties and passed to the [ExportToCsv](#) method which will export the properties to CSV format.

```

`ts
public IActionResult ServerSideExporting()
{
 var order = TreeData.GetDefaultData();
 ViewBag.dataSource = order;
 return View();
}

public IActionResult CsvExport(string treeGridModel)

```

```

{
if (treeGridModel == null)
{
return View();
}
TreeGridExcelExport exp = new TreeGridExcelExport();
Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
return exp.ExportToCsv<TreeData>(gridProperty, TreeData.GetDefaultData());
}

private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
(Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));

TreeGridColumnModel cols =
(TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
typeof(TreeGridColumnModel));

TreeGridModel.Columns = cols.columns;
return TreeGridModel;
}

public class TreeGridColumnModel
{
public List<TreeGridColumn> columns { get; set; }
}
`ts
import { TreeGrid, Toolbar } from '@syncfusion/ej2-treegrid';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
TreeGrid.Inject(Toolbar);
let data: DataManager = new DataManager({
url: "Home/DataSource",
adaptor: new UrlAdaptor
});
let treegrid: TreeGrid = new TreeGrid({

```

```

dataSource: data,
parentIdMapping: 'ParentItem',
hasChildMapping: 'isParent',
toolbar: ['CsvExport'],
columns: [
{ field: 'TaskID', headerText: 'Task ID', textAlign: 'Right', width: 90 },
{ field: 'TaskName', headerText: 'Task Name', width: 180 },
{ field: 'StartDate', headerText: 'Start Date', textAlign: 'Right', width: 90, format: { skeleton: 'yMd', type: 'date' } },
{ field: 'Duration', headerText: 'Duration', width: 80, textAlign: 'Right' }
],
height: 265
});
treegrid.appendTo('#TreeGrid');
treegrid.toolbarClick = (args: Object) => {
if (args['item'].id === 'TreeGridgridcontrolcsvexport') {
treegrid.serverCsvExport("Home/CsvExport");
}
}
,

```

*Rotate a header text to a certain degree in the exported tree grid on the server side*

The Tree Grid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported Excel file. To achieve this requirement, use the `ExcelHeaderCellRendering` event of the tree grid.

The `ExcelHeaderCellRendering` will be triggered when creating a column header for the excel document to be exported in the server side. Customize the column header in this event.

In the following demo, using the `HeaderCellRotate` method of the `TreeGridExcelExport` class in the `ExcelHeaderCellRendering` event, you can rotate the header text of the column header in the excel exported document.

```

`ts
public IActionResult ExcelExport(string treeGridModel)
{
if (treeGridModel == null)
{
return View();
}
}

```

```

}
TreeGridExcelExport exp = new TreeGridExcelExport();
Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
gridProperty.ExcelHeaderCellRendering = ExcelHeaderQueryCellInfo;
return (ActionResult)exp.ExcelExport<TreeGridItems>(gridProperty, TreeGridItems.GetDefaultData());
}
private void ExcelHeaderQueryCellInfo(object excel)
{
 Syncfusion.EJ2.TreeGridExport.ExcelHeaderCellEventArgs name =
 (Syncfusion.EJ2.TreeGridExport.ExcelHeaderCellEventArgs)excel;
 List<string> headerValues = new List<string>();
 headerValues.Add(name.Column.HeaderText);
 var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
 TreeGridExcelExport exp = new TreeGridExcelExport();
 var size = exp.ExcelTextSize(name.Style.Font.FontName, (float)name.Style.Font.Size, longestString);
 name.Cell.RowHeight = size.Width;
 exp.HeaderCellRotate(name, 45); // Give the rotate degree value by the user.
 name.Style.Borders.LineStyle = Syncfusion.XlsIO.ExcelLineStyle.None;
}

```

## Global local in EJ2 JavaScript Treegrid control

### Localization

The [Localization](#) library allows you to localize default text content of the TreeGrid. The treegrid component has static text on some features (like toolbar area text, filter menu text, pager information text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the

[locale](#) value and translation object.

The following list of properties and its values are used in the treegrid.

Locale keywords | Text

EmptyRecord | No records to display

True | true

False | false

ExpandAll | Expand All

CollapseAll | Collapse All

RowIndent | Indent



RowOutdent | Outdent

InvalidFilterMessage | Invalid Filter Data

FilterbarTitle | \s filter bar cell

Add | Add

Edit | Edit

Cancel | Cancel

Update | Update

Delete | Delete

Print | Print

Pdfexport | PDF Export

Excelexport | Excel Export

Wordexport | Word Export

Csvexport | CSV Export

Search | Search

Save | Save

EditOperationAlert | No records selected for edit operation

DeleteOperationAlert | No records selected for delete operation

SaveButton | Save

OKButton | OK

CancelButton | Cancel

EditFormTitle | Details of

AddFormTitle | Add New Record

ConfirmDelete | Are you sure you want to Delete Record?

SearchColumns | search columns

Matches | No Matches Found

FilterButton | Filter

ClearButton | Clear

StartsWith | Starts With

EndsWith | Ends With

Contains | Contains

Equal | Equal

NotEqual | Not Equal

LessThan | Less Than

LessThanOrEqualTo | Less Than Or Equal  
GreaterThan | Greater Than  
GreaterThanOrEqualTo | Greater Than Or Equal  
ChooseDate | Choose a Date  
EnterValue | Enter the value  
autoFitAll | Auto Fit all columns  
autoFit | Auto Fit this column  
Export | Export  
FirstPage | First Page  
LastPage | Last Page  
PreviousPage | Previous Page  
NextPage | Next Page  
SortAscending | Sort Ascending  
SortDescending | Sort Descending  
EditRecord | Edit Record  
DeleteRecord | Delete Record  
Above | Above  
Below | Below  
AddRow | Add Row  
FilterMenu | Filter  
SelectAll | Select All  
Blanks | Blanks  
FilterTrue | True  
FilterFalse | False  
NoResult | No Matches Found  
ClearFilter | Clear Filter  
NumberFilter | Number Filters  
TextFilter | Text Filters  
DateFilter | Date Filters  
MatchCase | Match Case  
Between | Between  
CustomFilter | Custom Filter  
CustomFilterPlaceholder | Enter the value

CustomFilterDatePlaceholder | Choose a date

AND | AND

OR | OR

ShowRowsWhere | Show rows where:

currentPageInfo | {0} of {1} pages

totalItemsInfo | ({0} items)

firstPageTooltip | Go to first page

lastPageTooltip | Go to last page

nextPageTooltip | Go to next page

previousPageTooltip | Go to previous page

nextPagerTooltip | Go to next pager

previousPagerTooltip | Go to previous pager

pagerDropDown | Items per page

pagerAllDropDown | Items

All | All

### *Loading translations*

To load translation object in an application, use [load](#) function of the `L10n` class.

The following example demonstrates the TreeGrid in `Deutsch` culture.

### **INDEX.TS**

```
import { L10n } from '@syncfusion/ej2-base';
import { TreeGrid, Toolbar, Page, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Toolbar, Filter, Page);
L10n.load({
 'de-DE': {
 'treegrid': {
 'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
 'Expand All': 'Alle erweitern',
 'Collapse All': 'Alles einklappen',
 'Print': 'Drucken',
 'Pdfexport': 'PDF-Export',
 'Excelexport': 'Excel-Export',
 'Wordexport': 'Word-Export',
 'FilterButton': 'Filter',
 'ClearButton': 'klar',
 'StartsWith': 'Beginnt mit',
 'EndsWith': 'Endet mit',
 'Contains': 'Enthält',
 'Equal': 'Gleich',
 'NotEqual': 'Nicht gleich',
 'LessThan': 'Weniger als',
 'LessThanOrEqual': 'Weniger als oder gleich',
 'GreaterThan': 'Größer als',
 }
 }
});
```

```

 "GreaterThanOrEqual": "Größer als oder gleich",
 "EnterValue": "Geben Sie den Wert ein",
 "FilterMenu": "Filter"
 },
 'pager': {
 'currentPageInfo': '{0} von {1} Seiten',
 'totalItemsInfo': '({0} Beiträge)',
 'firstPageTooltip': 'Zur ersten Seite',
 'lastPageTooltip': 'Zur letzten Seite',
 'nextPageTooltip': 'Zur nächsten Seite',
 'previousPageTooltip': 'Zurück zur letzten Seit',
 'nextPagerTooltip': 'Zum nächsten Pager',
 'previousPagerTooltip': 'Zum vorherigen Pager'
 },
 "dropdowns": {
 "noRecordsTemplate": "Keine Aufzeichnungen gefunden"
 },
 "datepicker": {
 "placeholder": "Wählen Sie ein Datum",
 "today": "heute"
 }
}
});
TreeGrid.Inject(Toolbar, Filter);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 locale: 'de-DE',
 childMapping: 'subtasks',
 toolbar: ['Print'],
 allowFiltering: true,
 filterSettings: {type: 'Menu'},
 height: 220,
 allowPaging: true,
 pageSettings: {pageSize: 7},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in treegrid component using format strings in the [columns.format](#).

### INDEX.TS

```

import { loadCldr, L10n, setCulture, setCurrencyCode } from
 '@syncfusion/ej2-base';
import * as currencies from './currencies.json';
import * as cagregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
import { TreeGrid, Page, Toolbar, Filter } from '@syncfusion/ej2-treegrid';
import { formatData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, Filter);
loadCldr(currencies, cagregorian, numbers, timeZoneNames, numberingSystems);
setCulture('de');
setCurrencyCode('EUR');
L10n.load({
 'de-DE': {
 'grid': {
 'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
 'Print': 'Drucken',
 'FilterButton': 'Filter',
 'ClearButton': 'klar',
 'StartsWith': 'Beginnt mit',
 'EndsWith': 'Endet mit',
 'Contains': 'Enthält',
 'Equal': 'Gleich',
 'NotEqual': 'Nicht gleich',
 'LessThan': 'Weniger als',
 'LessThanOrEqual': 'Weniger als oder gleich',
 'GreaterThan': 'Größer als',
 'GreaterThanOrEqual': 'Größer als oder gleich',
 'EnterValue': 'Geben Sie den Wert ein',
 'FilterMenu': 'Filter'
 },
 'pager': {
 'currentPageInfo': '{0} von {1} Seiten',
 'totalItemsInfo': '({0} Beiträge)',
 'firstPageTooltip': 'Zur ersten Seite',
 'lastPageTooltip': 'Zur letzten Seite',
 'nextPageTooltip': 'Zur nächsten Seite',
 'previousPageTooltip': 'Zurück zur letzten Seit',
 'nextPagerTooltip': 'Zum nächsten Pager',
 'previousPagerTooltip': 'Zum vorherigen Pager'
 },
 'dropdowns': {
 'noRecordsTemplate': 'Keine Aufzeichnungen gefunden'
 },
 'datepicker': {
 'placeholder': 'Wählen Sie ein Datum',
 'today': 'heute'
 }
 }
});
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: formatData,
 locale: 'de-DE',
 childMapping: 'subtasks',
 toolbar: ['Print'],
 allowFiltering: true,

```

```

 filterSettings: {type: 'Menu'},
 height: 220,
 allowPaging: true,
 pageSettings: {pageSize: 7},
 treeColumnIndex: 1,
 columns: [
 { field: 'orderId', headerText: 'Order ID', textAlign: 'Right',
width: 90 },
 { field: 'orderName', headerText: 'Order Name', textAlign:
'Left', width: 180 },
 { field: 'price', headerText: 'Price', textAlign: 'Right',
width: 80, format: {
 format: 'C2', useGrouping: false,
 minimumSignificantDigits: 1, maximumSignificantDigits: 3,
currency: 'EUR'
 }, type: 'number' },
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* In the above sample, **Price** column is formatted by **NumberFormatOptions**.

\* By default, **locale** value is **en-US**. If you want to change the **en-US** culture to a different culture, you have to change the **locale** accordingly.

### Right to left (RTL)

RTL provides an option to switch the text direction and layout of the TreeGrid component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Grid, set the **enableRtl** to true.

### INDEX.TS

```

import { L10n } from '@syncfusion/ej2-base';
import { TreeGrid, Page, Toolbar, Filter } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Toolbar, Filter);
L10n.load({
 'ar-AE': {
 'treegrid': {
 "EmptyRecord": "لا سجلات لعرضها",
 "Print": "طباعة",
 "FilterButton": "منقي",
 "ClearButton": "واضح",
 "StartsWith": "ابدا ب",
 "EndsWith": "ينتهي مع",
 "Contains": "يحتوي على",
 "Equal": "مساو",
 "NotEqual": "غير متساوي",
 "LessThan": "أقل من",
 "LessThanOrEqual": "اصغر من أو يساوي",
 "GreaterThan": "أكثر من",
 "GreaterThanOrEqual": "أكبر من أو يساوي",
 "ChooseDate": "اختر تاريخا",
 "EnterValue": "أدخل القيمة",

```



```

 "FilterMenu": "منقي"
 },
 'pager': {
 'currentPageInfo': '{0} صفحة 1 {من}',
 'totalItemsInfo': '({0} العناصر)',
 'firstPageTooltip': 'انتقل إلى الصفحة الأولى',
 'lastPageTooltip': 'انتقل إلى الصفحة الأخيرة',
 'nextPageTooltip': 'انتقل إلى الصفحة التالية',
 'previousPageTooltip': 'انتقل إلى الصفحة السابقة',
 'nextPagerTooltip': 'الذهاب إلى بيجر المقبل',
 'previousPagerTooltip': 'الذهاب إلى بيجر السابقة'
 },
 "dropdowns": {
 "noRecordsTemplate": "لا توجد سجلات"
 },
 "datepicker": {
 "placeholder": "اختر تاريخا",
 "today": "اليوم"
 }
}
});
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 locale: 'ar-AE',
 enableRtl: true,
 childMapping: 'subtasks',
 toolbar: ['Print'],
 allowFiltering: true,
 filterSettings: {type: 'Menu'},
 height: 210,
 allowPaging: true,
 pageSettings: {pageSize: 7},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Right' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

See Also

- [Internationalization](#)
- [Localization](#)

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

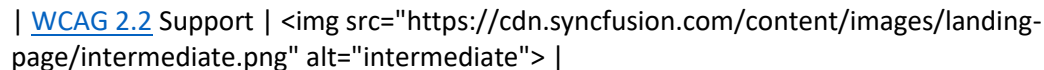
### Accessibility in EJ2 JavaScript Treegrid control

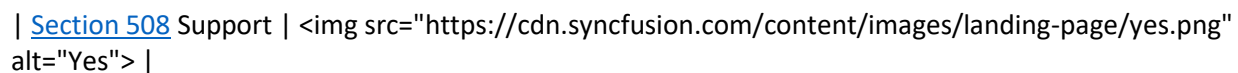
The Tree Grid component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

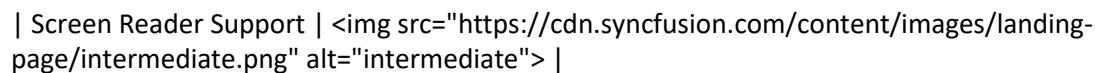
The accessibility compliance for the Tree Grid component is outlined below.

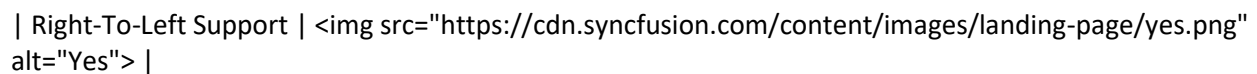
| Accessibility Criteria | Compatibility |

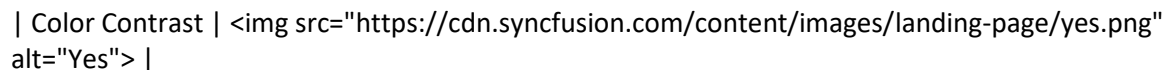
| -- | -- |

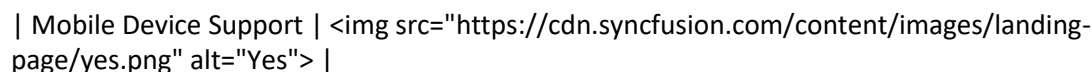
| [WCAG 2.2](#) Support |  |

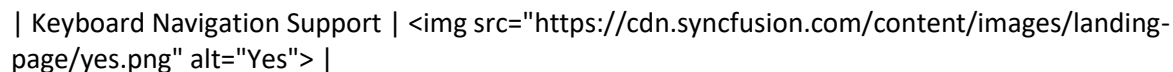
| [Section 508](#) Support |  |

| Screen Reader Support |  |

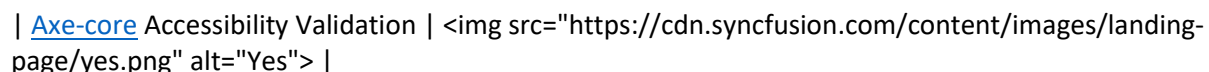
| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### WAI-ARIA attributes

The Tree Grid component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Tree Grid component:

Attributes	Purpose
---	---
<code>role=treegrid</code>	Used to convey a significant and contextual message to the user.
<code>aria-selected</code>	Accurately reflect the selection state, whether it's single-select or multi-select.
<code>aria-expanded</code>	It can be used to show whether a node is expanded or collapsed, making it easier for screen reader users to navigate and understand the hierarchy.
<code>aria-sort</code>	Indicate the current sorting order of a table column for users with disabilities, facilitating accessible data presentation and interaction.
<code>aria-busy</code>	Loading state to improve accessibility for users, particularly those relying on screen readers.
<code>aria-invalid</code>	To indicate whether the user's input in a form field is valid or invalid, aiding users, including those with disabilities, in understanding and correcting their input.
<code>aria-grabbed</code>	Provides accessibility information for users interacting with draggable elements.
<code>aria-owns</code>	Establishing relationships between an element and the elements it owns or controls.
<code>aria-label</code>	Provides an accessible name for the close icon.

### Keyboard interaction

The Tree Grid component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Tree Grid component.

Interaction Keys | Description

<code>PageDown</code>	Goes to the next page.
<code>PageUp</code>	Goes to the previous page.
<code>Ctrl + Alt + PageDown</code>	Goes to the last page.
<code>Ctrl + Alt + PageUp</code>	Goes to the first page.
<code>Alt + PageDown</code>	Goes to the next page.
<code>Alt + PageUp</code>	Goes to the previous page.
<code>Home</code>	Goes to the first cell.
<code>End</code>	Goes to the last cell.
<code>Ctrl + Home</code>	Goes to the first row.
<code>Ctrl + End</code>	Goes to the last row.

DownArrow | Moves the cell focus downward.

UpArrow | Moves the cell focus upward.

LeftArrow | Moves the cell focus left side.

RightArrow | Moves the cell focus right side.

Shift + DownArrow | Extends the row/cell selection downwards.

Shift + UpArrow | Extends the row/cell selection upwards.

Shift + LeftArrow | Extends the cell selection to the left side.

Shift + RightArrow | Extends the cell selection to the right side.

Enter | Moves the row/cell selection downward. If current cell is in edit state, then completes the editing. If the current cell is a header then performs sorting.

Shift + Enter | Moves the row/cell selection upward. If the current cell is a header then clears sorting for the selected column.

Ctrl + Enter | If the current cell is a header then performs multi-sorting.

Tab | Moves the cell selection right side.

Shift + Tab | Moves the cell selection left side.

Esc | Deselects all the rows/cells.

Ctrl + A | Selects all the rows/cells.

UpArrow | Moves up a row/cell selection.

DownArrow | Moves down a row/cell selection.

RightArrow | Moves to the right cell selection.

LeftArrow | Moves to the left cell selection.

Ctrl + Shift + DownArrow | Expands the selected group.

Ctrl + DownArrow | Expands all the visible groups.

Ctrl + Shift + UpArrow | Collapses the selected group.

Ctrl + UpArrow | Collapses all the visible groups.

Ctrl + P | Prints the TreeGrid.

### Ensuring accessibility

The Tree Grid component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Tree Grid component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Tree Grid component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Clipboard in EJ2 JavaScript Treegrid control

The clipboard provides an option to copy selected rows or cells data into the clipboard.

The following list of keyboard shortcuts is supported in the Tree Grid to copy selected rows or cells data into the clipboard.

Interaction keys | Description

**Ctrl + C | Copy selected rows or cells data into clipboard.**

**Ctrl + Shift + H | Copy selected rows or cells data with header into clipboard.**

#### INDEX.TS

```
import { TreeGrid, Page, Selection } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Selection);
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 allowSelection: true,
 allowPaging: true,
 pageSettings: { pageSize: 10 },
 selectionSettings: { type: 'Multiple', mode: 'Row' },
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80,
textAlign: 'Right' },
],
 height: 260
 });
treegrid.appendTo('#TreeGrid');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Typescript Tree Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<!-- Syncfusion Essential JS 2 Styles -->
<link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="TreeGrid"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Copy to clipboard by external buttons

To copy selected rows or cells data into the clipboard with help of external buttons, you need to invoke the [copy](#) method.

### INDEX.TS

```

import { TreeGrid, Page, Selection } from '@syncfusion/ej2-treegrid';
import { Button } from '@syncfusion/ej2-buttons';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Selection);
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 allowSelection: true,
 selectionSettings: { type: 'Multiple', mode: 'Row' },
 allowPaging: true,
 pageSettings: { pageSize: 10 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80,
textAlign: 'Right' },

```

```

],
 height: 230
 });
 treegrid.appendTo('#TreeGrid');
 let copyBtn: Button = new Button();
 copyBtn.appendTo('#copy');
 document.getElementById('copy').addEventListener('click', () => {
 treegrid.copy();
 });
 let copyHeaderBtn: Button = new Button();
 copyHeaderBtn.appendTo('#copyHeader');
 document.getElementById('copyHeader').addEventListener('click', () => {
 treegrid.copy(true);
 });

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="copy">Copy</button>
 <button id="copyHeader">Copy With Header</button>
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Copy Hierarchy Modes

Tree Grid provides support for a set of copy modes with [copyHierarchyMode](#) property.

The below are the type of filter mode available in TreeGrid.



- **Parent** : This is the default copy hierarchy mode in Tree Grid. Clipboard value have the selected records with its parent records. If the selected records not have any parent record then the selected record will be in clipboard.
- **Child** : Clipboard value have the selected records with its child record. If the selected records do not have any child record then the selected records will be in clipboard.
- **Both** : Clipboard value have the selected records with its both parent and child record. If the selected records do not have any parent and child record then the selected records alone in clipboard.
- **None** : Only the Selected records will be in clipboard.

## INDEX.TS

```
import { TreeGrid, Page, Selection } from '@syncfusion/ej2-treegrid';
import { DropDownList, ChangeEventArgs } from '@syncfusion/ej2-dropdowns';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Selection);
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 copyHierarchyMode: 'Parent',
 allowSelection: true,
 selectionSettings: { type: 'Multiple', mode: 'Row' },
 allowPaging: true,
 pageSettings: { pageSize: 10 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80,
textAlign: 'Right' },
],
 height: 230
 });
treegrid.appendTo('#TreeGrid');
let dropdownMode: DropDownList = new DropDownList({
 dataSource: [
 { id: 'Parent', mode: 'Parent' },
 { id: 'Child', mode: 'Child' },
 { id: 'Both', mode: 'Both' },
 { id: 'None', mode: 'None' },
],
 fields: { text: 'mode', value: 'id' },
 value: 'Parent',
 change: (e: ChangeEventArgs) => {
 let mode: any = <string>e.value;
 treeGridObj.copyHierarchyMode = mode;
 }
});
```

```
dropDownMode.appendTo('#mode');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div>
 <div style="padding-top: 7px; display: inline-block">Hierarchy
Mode</div>
 <div style="display: inline-block">
 <input type="text" id="mode">
 </div>
 </div>
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

### Limitations of Copy Functionality

- Only current view records will be available in copy clipboard.

### AutoFill

AutoFill Feature allows you to copy the data of selected cells and paste it to another cells by just dragging the autofill icon of the selected cells up to required cells. This feature is enabled by defining `enableAutoFill` property as true.

## INDEX.TS

```
import { TreeGrid, Page, Selection, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Selection, Edit, Toolbar);
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 enableAutoFill: true,
 enableHover: false,
 allowSelection: true,
 toolbar: ['Add', 'Update', 'Cancel'],
 selectionSettings: { type: 'Multiple', mode: 'Cell',
cellSelectionMode: 'Box' },
 editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Batch' },
 allowPaging: true,
 pageSettings: { pageSize: 10 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true,
width: 70, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80,
textAlign: 'Right' },
],
 height: 220
 });
treegrid.appendTo('#TreeGrid');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```

 <div id="TreeGrid"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

\* If **enableAutoFill** is set to true, then the autofill icon will be displayed on cell selection to copy cells.

\* It requires the selection **mode** to be **Cell**, **cellSelectionMode** to be **Box** and also Batch Editing should be enabled.

#### *Limitations of AutoFill*

- Since the string values are not parsed to number and date type, so when the selected string type cells are dragged to number type cells then it will display as **NaN**. For date type cells, when the selected string type cells are dragged to date type cells then it will display as an **empty cell**.
- Linear series and the sequential data generations are not supported in this autofill feature.

#### *Paste*

You can able to copy the content of a cell or a group of cells by selecting the cells and pressing Ctrl + C shortcut key and paste it to another set of cells by selecting the cells and pressing Ctrl + V shortcut key.

#### INDEX.TS

```

import { TreeGrid, Page, Selection, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page, Selection, Edit, Toolbar);
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 enableHover: false,
 allowSelection: true,
 toolbar: ['Add', 'Update', 'Cancel'],
 selectionSettings: { type: 'Multiple', mode: 'Cell',
cellSelectionMode: 'Box' },
 editSettings: { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Batch' },
 allowPaging: true,
 pageSettings: { pageSize: 10 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70, textAlign:
'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },

```

```

 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'progress', headerText: 'Progress', width: 80,
textAlign: 'Right' },
],
 height: 220
});
treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="TreeGrid"></div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

To perform paste functionality, it requires the selection **mode** to be **Cell**, **cellSelectionMode** to be **Box** and also Batch Editing should be enabled.

### *Limitations of Paste Functionality*

- Since the string values are not parsed to number and date type, so when the copied string type cells are pasted to number type cells then it will display as **NaN**. For date type cells, when the copied string format cells are pasted to date type cells then it will display as an **empty cell**.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Context menu in EJ2 JavaScript Treegrid control

The TreeGrid has options to show the context menu when right clicked on it. To enable this feature, you need to define either default or custom item in the [contextMenuItems](#).

To use the context menu, inject the **ContextMenu** module in the treegrid.

The default items are in the following table.

Items	Description
<b>AutoFit</b>	Auto fit the current column.
<b>AutoFitAll</b>	Auto fit all columns.
<b>Edit</b>	Edit the current record.
<b>Delete</b>	Delete the current record.
<b>Save</b>	Save the edited record.
<b>Cancel</b>	Cancel the edited state.
<b>PdfExport</b>	Export the treegrid data as Pdf document.
<b>ExcelExport</b>	Export the treegrid data as Excel document.
<b>CsvExport</b>	Export the treegrid data as CSV document.
<b>SortAscending</b>	Sort the current column in ascending order.
<b>SortDescending</b>	Sort the current column in descending order.
<b>FirstPage</b>	Go to the first page.
<b>PrevPage</b>	Go to the previous page.
<b>LastPage</b>	Go to the last page.
<b>NextPage</b>	Go to the next page.
<b>AddRow</b>	Add new row to the treegrid.
<b>Indent</b>	Indents the record to one level of hierarchy.
<b>Outdent</b>	Outdents the record to one level of hierarchy.

### INDEX.TS

```
import { TreeGrid, Resize, Sort, ContextMenu, Edit, Page, PdfExport,
ExcelExport, RowDD } from '@syncfusion/ej2-treegrid';
import { sampleData } from '../datasource.ts';
TreeGrid.Inject(Resize, Sort, Edit, ContextMenu, Page, PdfExport,
ExcelExport, RowDD);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
```

```

 allowSorting: true,
 allowResizing: true,
 allowPaging: true,
 pageSettings: {pageSize: 7},
 editSettings: {allowEditing: true, allowAdding: true, allowDeleting:
true, mode:"Row"},
 allowPdfExport: true,
 allowExcelExport: true,
 contextMenuItems: ['AutoFit', 'AutoFitAll', 'SortAscending',
'SortDescending', 'Edit', 'Delete', 'Save', 'Cancel',
'PdfExport', 'ExcelExport', 'CsvExport'
'FirstPage', 'PrevPage', 'LastPage', 'NextPage', 'Indent', 'Outdent'],
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey:
true, width: 90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
editType: 'datePickeredit', textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
editType: 'numericedit', textAlign: 'Right' }
]
 });
 treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Custom context menu items

The custom context menu items can be added by defining the [contextMenuItems](#) as a collection of [contextMenuItemModel](#).

Actions for this customized items can be defined in the [contextMenuClick](#) event.

In the below sample, we have shown context menu item for parent rows to expand or collapse child rows.

### INDEX.TS

```

import { TreeGrid, ContextMenu, Page } from '@syncfusion/ej2-treegrid';
import { getValue, isNullOrUndefined } from '@syncfusion/ej2-base';
import { sampleData } from './datasource.ts';
import { BeforeOpenCloseEventArgs } from '@syncfusion/ej2-inputs';
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
TreeGrid.Inject(ContextMenu, Page);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 pageSettings: { pageSize: 7 },
 contextMenuItems: [
 { text: 'Collapse the Row', target: '.e-content', id:
'collapserow'},

```



```

 {text: 'Expand the Row', target: '.e-content', id:
'expandrow'}
],
 contextMenuClick: (args?: MenuEventArgs) => {
 treeGridObj.getColumnByField('taskID');
 if (args.item.id === 'collapserow') {
 treeGridObj.collapseRow(<HTMLTableRowElement>(treeGridObj.getSelectedRows()[0]));
 } else {
 treeGridObj.expandRow(<HTMLTableRowElement>(treeGridObj.getSelectedRows()[0]));
 }
 },
 contextMenuOpen: (arg?: BeforeOpenCloseEventArgs) => {
 let elem: Element = arg.event.target as Element;
 let uid: string = elem.closest('.e-row').getAttribute('data-uid');
 if (isNullOrUndefined(getValue('hasChildRecords', treeGridObj.grid.getRowObjectFromUID(uid).data))) {
 arg.cancel = true;
 } else {
 let flag: boolean = getValue('expanded', treeGridObj.grid.getRowObjectFromUID(uid).data);
 let val: string = flag ? 'none' : 'block';

 document.querySelectorAll('li#expandrow')[0].setAttribute('style', 'display: ' + val + ';');
 val = !flag ? 'none' : 'block';

 document.querySelectorAll('li#collapserow')[0].setAttribute('style', 'display: ' + val + ';');
 }
 },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
 textAlign: 'Left' },
 {
 field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Typescript Grid Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Enable and disable context menu items dynamically

You can enable and disable the context menu items using the [enableItems](#) method in [contextMenuOpen](#) event.

**INDEX.TS**

```

import { TreeGrid, ContextMenu, Page, Edit } from '@syncfusion/ej2-treegrid';
import { getValue, isNullOrUndefined } from '@syncfusion/ej2-base';
import { sampleData } from './datasource.ts';
import { BeforeOpenCloseEventArgs } from '@syncfusion/ej2-inputs';
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
TreeGrid.Inject(ContextMenu, Page, Edit);
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,
 editSettings: {
 allowEditing: true,
 allowAdding: true,
 allowDeleting: true,
 mode: 'Row',
 },
 pageSettings: {pageSize: 7},
 contextMenuItems: [
 { text: 'Edit Record', target: '.e-content', id: 'Edit_record' },
 { text: 'Delete Record', target: '.e-content', id: 'Delete_record' }
],
 contextMenuClick: (args?: MenuEventArgs) => {
 if(args.element.innerHTML == "Edit Record"){
 treeGridObj.startEdit(args.rowInfo.row);
 }
 else if(args.element.innerHTML == "Delete Record"){
 treeGridObj.deleteRecord(args.rowInfo.row);
 }
 },
 contextMenuOpen: (args?: BeforeOpenCloseEventArgs) => {
 if (args.rowInfo.rowData.hasChildRecords == true){

treeGridObj.grid.contextMenuModule.contextMenu.enableItems(['Edit
Record'],true);//Enable edit

treeGridObj.grid.contextMenuModule.contextMenu.enableItems(['Delete
Record'],false);//Disable delete
 } else {

treeGridObj.grid.contextMenuModule.contextMenu.enableItems(['Edit
Record'],false);//Disable edit

treeGridObj.grid.contextMenuModule.contextMenu.enableItems(['Delete
Record'],true);//Enable edit
 }
 },
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180,
textAlign: 'Left' },
],

```

```

 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd'
 },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">

```

```

 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can hide or show an item in context menu for specific area inside of treegrid by defining the [target](#) property.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Adaptive in EJ2 JavaScript Treegrid control

The Tree Grid user interface (UI) was redesigned to provide an optimal viewing experience and improve usability on small screens. This interface will render the filter, edit dialog and other features adaptively. For example, Filtering opens the UI for user in a pop-up occupying the entire screen.

To make the tree grid adaptive, set the [enableAdaptiveUI](#) to true.

### INDEX.TS

```

import { TreeGrid, Filter, Sort, Edit, Toolbar, Page } from
 '@syncfusion/ej2-treegrid';
import { Browser } from '@syncfusion/ej2-base';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Filter, Sort, Edit, Toolbar, Page);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 enableAdaptiveUI: true,
 allowPaging: true,
 allowSorting: true,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 allowFiltering: true,
 filterSettings: { type: 'Excel' },
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel', 'Search'],
 editSettings: { allowAdding: true, allowEditing: true, allowDeleting:
true, mode: 'Dialog' },
 height: '100%',
 load: () => {
 treegrid.grid.adaptiveDlgTarget =
document.getElementsByClassName('e-mobile-content')[0] as HTMLElement;
 },
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
135, validationRules: { required: true, number: true } },
 { field: 'taskName', headerText: 'Task Name', width: 280, editType:
"dropdownedit", validationRules: { required: true } },
 { field: 'duration', headerText: 'Duration', filter: { type : 'Menu'
}, width: 140, validationRules: { required: true } },

```

```

 { field: 'progress', headerText: 'Progress', width: 145 ,
validationRules: { required: true } },
]
});
treegrid.appendTo('#adaptivebrowser');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <style>
 #container {
 visibility: hidden;
 }

 #loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;

```

```
top: 45%;
width: 30%;
}

/* The device with borders */
.e-mobile-layout {
 position: relative;
 width: 360px;
 height: 640px;
 margin: auto;
 border: 16px #f4f4f4 solid;
 border-top-width: 60px;
 border-bottom-width: 60px;
 border-radius: 36px;
 box-shadow: 0 0px 2px rgb(144 144 144), 0 0px 10px rgb(0 0 0 / 16%);
}

/* The horizontal line on the top of the device */
.e-mobile-layout:before {
 content: '';
 display: block;
 width: 60px;
 height: 5px;
 position: absolute;
 top: -30px;
 left: 50%;
 transform: translate(-50%, -50%);
 background: #ebebeb;
 border-radius: 10px;
}

/* The circle on the bottom of the device */
.e-mobile-layout:after {
 content: '';
 display: block;
 width: 35px;
 height: 35px;
 position: absolute;
 left: 50%;
 bottom: -65px;
 transform: translate(-50%, -50%);
 background: #e8e8e8;
 border-radius: 50%;
}

/* The screen (or content) of the device */
.e-mobile-layout .e-mobile-content {
 overflow-x: hidden;
 height: 100%;
 background: white;
 border: 0px solid #dddddd;
}

.highcontrast .e-mobile-layout {
 border: 16px #000000 solid;
 border-top-width: 60px;
 border-bottom-width: 60px;
```

```

 box-shadow: -1px 2px white, -2px -2px white, 2px -2px white, 2px 1px
white;
 }

 .e-responsive-dialog {
 box-shadow: none;
 border: 1px solid #dddddd;
 }

 /* Render the mobile pager by default */
 @media (max-width: 3840px) {
 .e-adaptive-demo .e-pager {
 padding: 13px 0;
 }
 .e-adaptive-demo .e-pager div.e-parentmsgbar {
 box-sizing: border-box;
 display: inline-block;
 float: initial;
 padding-bottom: 0;
 padding-right: 0;
 padding-top: 0;
 text-align: center;
 vertical-align: top;
 width: calc(60% - 48px);
 }
 .e-adaptive-demo .e-pager .e-pagesizes {
 display: none;
 }
 .e-adaptive-demo .e-pager .e-pagecountmsg {
 display: none;
 }
 .e-adaptive-demo .e-pager .e-pagercontainer {
 display: none;
 }
 .e-adaptive-demo .e-pager .e-icons {
 font-size: 11px;
 }
 .e-adaptive-demo .e-pager .e-mfirst,
 .e-adaptive-demo .e-pager .e-mprev,
 .e-adaptive-demo .e-pager .e-mnext,
 .e-adaptive-demo .e-pager .e-mlast {
 border: 0;
 box-sizing: border-box;
 display: inline-block;
 padding: 1% 5%;
 }
 .e-adaptive-demo .e-pager .e-mfirst {
 margin-right: 4px;
 text-align: right;
 width: calc(10% + 11px);
 }
 .e-adaptive-demo .e-pager .e-mprev {
 margin: 0 4px;
 text-align: right;
 width: 10%;
 }
 .e-adaptive-demo .e-pager .e-mnext {

```



```

 margin: 0 4px;
 text-align: left;
 width: 10%;
 }
 .e-adaptive-demo .e-pager .e-mlast {
 margin-left: 4px;
 text-align: left;
 width: calc(10% + 11px);
 }
 .e-adaptive-demo .e-bigger .e-pager,
 .e-adaptive-demo .e-pager.e-bigger {
 padding: 19px 0;
 }
 .e-adaptive-demo .e-bigger .e-pager.e-rtl div.e-parentmsgbar,
 .e-adaptive-demo .e-pager.e-bigger.e-rtl div.e-parentmsgbar {
 margin-right: 0;
 }
 .e-adaptive-demo .e-bigger .e-pager div.e-parentmsgbar,
 .e-adaptive-demo .e-pager.e-bigger div.e-parentmsgbar {
 padding: 0;
 }
 }
 }

 .e-dlg-target.e-scroll-disabled {
 overflow: auto !important;
 }
}

</style><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

<body>

 <div id="container" class="e-adaptive-demo e-bigger">
 <div class="col-lg-9 control-section">
 <div class="content-wrapper e-bigger e-adaptive-
demo">

 <div class="e-mobile-layout">
 <div class="e-mobile-content">
 <div
id="adaptivebrowser"></div>

 </div>
 </div>
 <div id="adaptivedevice"></div>
 </div>
 </div>

 <div class="datalink">Source:
 <a
href="https://en.wikipedia.org/wiki/List_of_Android_smartphones"
target="_blank">Wikipedia: List of Android smartphones
 </div>

```

```

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Please refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to learn how to present and manipulate data.

### Loading animation in EJ2 JavaScript Treegrid control

The Tree Grid displays a loading indicator while the data is being fetched and bound to the tree grid during initial rendering, refreshing, and after performing any tree grid actions like sorting, paging and more.

The tree grid supports two indicator types, which can be enabled by setting the `loadingIndicator.indicatorType` property to `Spinner` or `Shimmer`. The default value of the indicator type is `Spinner`.

In the following sample, the Shimmer indicator is displayed while the tree grid is loading and refreshing when using the remote data.

#### INDEX.TS

```

import { TreeGrid, Page, Sort } from '@syncfusion/ej2-treegrid';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
TreeGrid.Inject(Page, Sort);
let data = new DataManager({
 url: 'https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData',
 adaptor: new WebApiAdaptor(),
 crossDomain: true
});
let treegrid: TreeGrid = new TreeGrid(
{
 dataSource: data,
 hasChildMapping: 'isParent',
 idMapping: 'TaskID',
 parentIdMapping: 'ParentItem',
 height: 400,
 treeColumnIndex: 1,
 allowPaging: true,
 allowSorting: true,
 loadingIndicator: { indicatorType: 'Shimmer' },
 columns: [
 { field: 'TaskID', headerText: 'Task ID', width: 120,
textAlign: 'Right' },
 { field: 'TaskName', headerText: 'Task Name', width: 240,
textAlign: 'Left' },
 { field: 'StartDate', headerText: 'Start Date', width: 140,
textAlign: 'Right', type: 'date', format: 'yMd' },

```

```

 { field: 'Duration', headerText: 'Duration', width: 130,
textAlign: 'Right' },
 { field: 'Progress', headerText: 'Progress', width: 130 },
],
 pageSettings: { pageCount: 3 }
});
treegrid.appendTo('#TreeGrid');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

 <style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 </style><script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Test Helpers

### Helpers in EJ2 JavaScript Treegrid control

Essential JS 2 components packages provide the testing helper APIs for testing the Essential JS 2 components. The following helper APIs are available for the Tree Grid component's testing helpers in Essential JS 2.

Helper Methods	Description
:-----: -----	
<code>getTreeGridElement</code>	This function helps you to get the treegrid element of the Tree Grid. <b>Syntax:</b> <code>curObj.getTreeGridElement()</code>
<code>getHeaderElement</code>	This function helps you to get the header element of the Tree Grid. <b>Syntax:</b> <code>curObj.getHeaderElement()</code>
<code>getContentElement</code>	This function helps you to get the content element of the Tree Grid. <b>Syntax:</b> <code>curObj.getContentElement()</code>
<code>getFooterElement</code>	This function helps you to get the footer element of the Tree Grid when Aggregates are enabled. <b>Syntax:</b> <code>curObj.getFooterElement()</code>
<code>getPagerElement</code>	This function helps you to get the pager element of the Tree Grid when paging feature is enabled. <b>Syntax:</b> <code>curObj.getPagerElement()</code>
<code>getFilterPopUpElement</code>	This function helps you to get the Filter Dialog element of the Columns in Tree Grid. <b>Syntax:</b> <code>curObj.getFilterPopUpElement()</code>
<code>getDialogElement</code>	This function helps you to get the Edit Dialog element of Tree Grid when the Edit Mode is Dialog. <b>Syntax:</b> <code>curObj.getDialogElement()</code>
<code>getCurrentPagerElement</code>	This function helps you to get the current page indicator element of the Tree Grid. <b>Syntax:</b> <code>curObj.getCurrentPagerElement()</code>
<code>getPagerDropDownElement</code>	This function helps you to get the page size dropdownlist element of the TreeGrid. <b>Syntax:</b> <code>curObj.getPagerDropDownElement()</code>
<code>getExpandedElements</code>	This function helps you to get all the expanded icon elements in the current page of Tree Grid. <b>Syntax:</b> <code>curObj.getExpandedElements()</code>
<code>getCollapsedElements</code>	This function helps you to get all the collapsed icon elements in the current page of Tree Grid. <b>Syntax:</b> <code>curObj.getCollapsedElements()</code>

### Cypress in EJ2 JavaScript Treegrid control

This section describes how to create a Cypress project and test the Tree Grid component using the Tree Grid Test Helper.

#### Setting up the Cypress Project

To set the cypress project, follow these steps to create a project:

- Open the command prompt and set up the root project directory.
- Run the **npm init** command.

```

D:\Development\test\cypress>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (cypress) cypressstesting
version: (1.0.0)
description: initializing the cypress installation
entry point: (index.js)
test command: npm run cypress: open
git repository:
keywords:
author:
license: (ISC)
About to write to D:\Development\test\cypress\package.json:
{
 "name": "cypressstesting",
 "version": "1.0.0",
 "description": "initializing the cypress installation",
 "main": "index.js",
 "scripts": {
 "test": "npm run cypress: open"
 },
 "author": "",
 "license": "ISC"
}

Is this OK? (yes) yes
D:\Development\test\cypress>

```

- Install Cypress to the project. Refer to the Cypress [installation link](#) to install.
- Start writing [your testcase](#).

### Installing syncfusion package

The test helpers are available in the Syncfusion default component packages. Here, Tree Grid component is tested. So, run the following command to install the Syncfusion Tree Grid component.

```
npm install @syncfusion/ej2-treegrid --save
```

### Importing Test Helper from the package

You can test the drop-down components by using the `TreeGridHelper` class that is imported from the `@syncfusion/ej2-treegrid` package. The helper class is available in the `helpers/e2e` folder in every Syncfusion components package.

- Create a `sample_spec.js` file in the `cypress/integration/` folder.
- Import the testinghelper class name which is to be tested in the created file.

*Syntax:*

```

`ts
import { TestHelperClassName } from '{ helperPath }'
,

```

*For example,*

```

`ts
import { TreeGridHelper } from '@syncfusion/ej2-treegrid/helpers/e2e'

```

,

### Initializing the test helper

After importing the testing helper, follow these steps:

- Set the website page URL or local sample path that you want to test in browser. Here, the demo site is used for example.

`ts

```
cy.visit('https://ej2.syncfusion.com/demos/tree-grid/default-paging/index.html')
```

,

- Initialize the helpers as follows.

`ts

```
var curTreeGrid = new TreeGridHelper('TreeGrid', cy.get)
```

,

Refer to the following screenshot for detailed helper class initiation.

```
describe('The Home Page', function(e){
 var treegridobj;
 it('on successful load', function(done){
 cy.visit("https://ej2.syncfusion.com/javascript/demos/#/material/tree-grid/default-paging.html").then(function(){
 treegridobj = new TreeGridHelper('TreeGrid', cy.get);
 done();
 });
 });
});
```

### Write test case using helper methods

After initializing the helpers, you can write the test cases. Following are some of the simple examples for helper functions with test cases.

#### Using getModel setModel

After the visiting page is loaded, write the test case to get and set the values.

For example,

If you need to set the element's value, then you can use this `setModel` helper function.

Similarly, the element's value is tested by the `getModel` helper function.

`ts

```
it('getting setting model', function () {
 curTreeGrid.setModel("allowPaging", false);
 curTreeGrid.getModel("allowPaging").should('eq', false);
});
```

,

#### Using Invoke function

If you want to invoke some functions, you can use the `invoke` helper function.

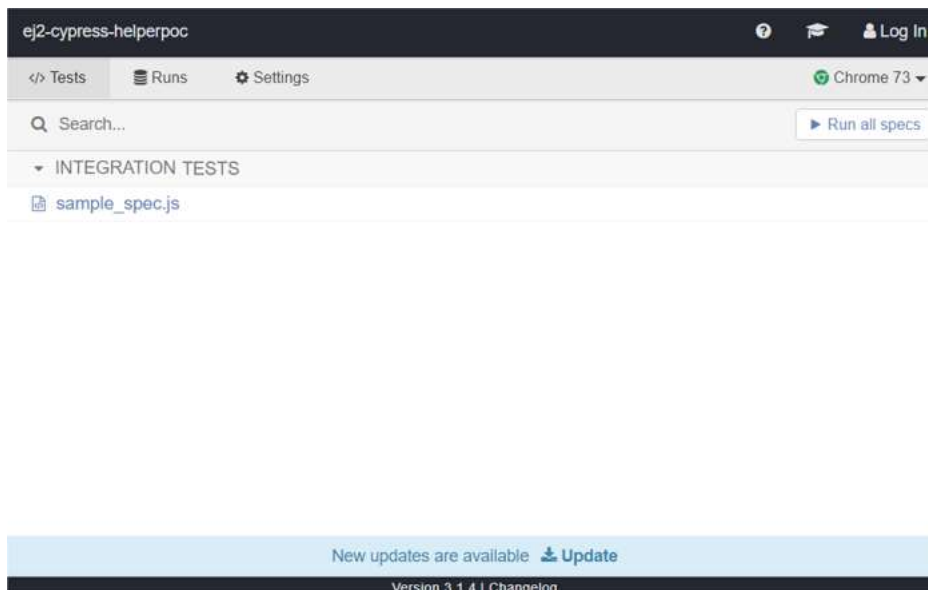
```
`ts
it('invoke function', function () {
 curTreeGrid.invoke("collapseAll");
});
`
```

Here, the `collapseAll` method is used to collapse all the parent records in the Tree Grid component.

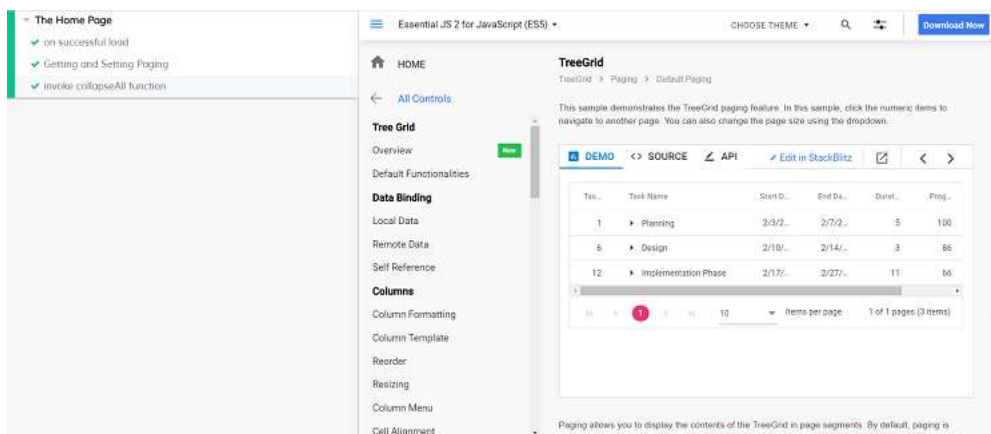
### Running Cypress

To run the Cypress, follow these steps:

- Run the `npm run cypress:open` terminal command from the root directory to open the Cypress test runner application. The Cypress application will open as follows.



- Now, the `sample_spec.js` file will show in the Cypress application window. Click that file to get the output window as follows.



Note: Refer to the GitHub samples for quick implementation and testing from [here](#).

### Treegrid styling in EJ2 JavaScript Treegrid control

To modify the TreeGrid appearance, you need to override the default CSS of treegrid. Please find the list of CSS classes and its corresponding section in treegrid. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Section | CSS class | Purpose of CSS class

**Root** | e-treegrid | This classes are in this root element (div) of the treegrid control.

**Header** | e-gridheader | This class is added in the root element of header element. In this class, You can override thin line between header and content of the treegrid.

| e-table | This class is added at 'table' of the treegrid header. This CSS class makes table width as 100 %.

| e-columnheader | This class is added at 'tr' of the treegrid header.

| e-headercell | This class is added in 'th' element of treegrid header. You can override background color of header and border color.

| e-headercelldiv | This class is add in div which present 'th' element in the header. we recommend you to use the e-headercelldiv to override skeleton of header.

**Body** | e-gridcontent | This class is added at root of body content. This is to override background color of the body.

| e-table | This class is added to table of content. This CSS class makes table width as 100 %.

| e-altrow | This class is added to alternate rows of treegrid. This is to override alternate row color of the treegrid.

| e-rowcell | This class is added to all cells in the treegrid. This is to override cells appearance and styling.

| e-groupcaption | This class is added to the 'td' of group caption which is to change the background color of caption cell.

| e-selectionbackground | This class is added to rowcell's of the treegrid. This is override selection.

| e-hover | This class adds to row of treegrid, while hovering the treegrid rows.

**Pager** | e-pager | This class is added to root element of the pager. This to change appearance of the background color and color of font.

| e-pagercontainer | This class is added to numeric items of the pager.

| e-parentmsgbar | This class is added to pager info of the pager.

**Summary** | e-gridfooter | This class is added to root of the summary div.

| e-summaryrow | This class is added to rows of treegrid summary.

| e-summarycell | This class is added to cells of summary row. This to override background color of summary.

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to knows how to present and manipulate data.



## How To

### Refresh the data source in EJ2 JavaScript Treegrid control

#### *How to refresh the datasource*

You can add/delete the datasource records through an external button. To reflect the datasource changes in Tree Grid, you need to assign the modified data to dataSource property.

Please follow the below steps to refresh the Tree Grid after datasource change.

#### Step 1:

Add/delete the data source record by using the following code.

```
`ts
treegrid.dataSource.unshift(data); // add a new record.
treegrid.dataSource.splice(selectedRow, 1); // delete a record.
`
```

#### Step 2:

Refresh the Tree Grid after the data source change by using the [refresh](#) method.

```
`ts
treegrid.refresh(); // refresh the Grid.
`
```

#### INDEX.TS

```
import { TreeGrid ,extendArray} from '@syncfusion/ej2-treegrid';
import { Button } from '@syncfusion/ej2-buttons';
import { projectData } from './datasource.ts';
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: 'TaskID',
 parentIdMapping: 'parentID',
 treeColumnIndex: 1,
 columns: [
 { field: 'TaskID', headerText: 'Task ID', width: 70, textAlign: 'Right' },
 { field: 'TaskName', headerText: 'Task Name', width: 100, textAlign: 'Left' },
 { field: 'StartDate', headerText: 'Start Date', width: 90, textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'EndDate', headerText: 'End Date', width: 90, textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'Duration', headerText: 'Duration', width: 90, textAlign: 'Right' },
 { field: 'Priority', headerText: 'Priority', width: 90, textAlign: 'Right' }
],
});
treegrid.appendTo('#TreeGrid');
let add: Button = new Button({}, '#add');
let dele: Button = new Button({}, '#delete');
document.getElementById('add').onclick = () => {
```

```

 let treegrid = document.getElementsByClassName("e-
treegrid")[0].ej2_instances[0]; //take treegrid instance here
 let dataSource = extendArray(treegrid.dataSource);
 dataSource.unshift({ TaskID: 99, TaskName: "New Data", StartDate: new
Date('09/07/2020'), Duration: 10, Priority: "High" }); // Add record
 treegrid.dataSource = dataSource; // Refresh the TreeGrid.
};
document.getElementById('delete').onclick = () => {
 let treegrid = document.getElementsByClassName("e-
treegrid")[0].ej2_instances[0]; //take treegrid instance here
 let selectedRow = treegrid.getSelectedRowIndex().length;
 let selectedRowIndex = treegrid.getSelectedRowIndex()[0];
 let dataSource = extendArray(treegrid.dataSource);
 if (selectedRow > 0) {
 dataSource.splice(selectedRowIndex, 1); // Delete record.
 }
 else {
 alert("No records selected for delete operation");
 }
 treegrid.dataSource = dataSource; // Refresh the TreeGrid.
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button class="e-flat" id="add">Add</button>
 <button class="e-flat" id="delete">Delete</button>
 <div id="TreeGrid"></div>
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

[Enable/disable treegrid and its actions in EJ2 JavaScript Treegrid control](#)

You can enable/disable the Tree Grid and its actions by applying/removing corresponding CSS styles.

To enable/disable the Tree Grid and its actions, follow the given steps:

#### Step 1:

Create CSS class with custom style to override the default style of Tree Grid.

```

,

.disabletreegrid {
 pointer-events: none;
 opacity: 0.4;
}

.wrapper {
 cursor: not-allowed;
}
,

```

#### Step 2:

Add/Remove the CSS class to the Tree Grid in the click event handler of Button.

```
`ts
document.getElementById('primarybtn').addEventListener('click', () => {
 if (treegrid.element.classList.contains('disabletreegrid')) {
 treegrid.element.classList.remove('disabletreegrid');
 document.getElementById("TreeGridParent").classList.remove('wrapper');
 }
 else {
 treegrid.element.classList.add('disabletreegrid');
 document.getElementById("TreeGridParent").classList.add('wrapper');
 }
});
`
```

In the below demo, the button click will enable/disable the Tree Grid and its actions.

#### INDEX.TS

```
import { TreeGrid, Edit, Toolbar } from '@syncfusion/ej2-treegrid';
import { Button } from '@syncfusion/ej2-buttons';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let button: Button = new Button();
button.appendTo('#primarybtn');
let treegrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 height: 220,
 toolbar: ['Add', 'Edit', 'Delete', 'Update', 'Cancel'],
 editSettings: { allowAdding: true, allowEditing: true, allowDeleting: true
},
 columns: [
 { field: 'taskID', headerText: 'Task ID', isPrimaryKey: true, width:
90, textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90, textAlign:
'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
'Right' }
],
});
treegrid.appendTo('#TreeGrid');
document.getElementById('primarybtn').addEventListener('click', () => {
 if (treegrid.element.classList.contains('disabletreegrid')) {
 treegrid.element.classList.remove('disabletreegrid');
 }
 document.getElementById("TreeGridParent").classList.remove('wrapper');
```

```

 }
 else {
 treegrid.element.classList.add('disabletreegrid');
 document.getElementById("TreeGridParent").classList.add('wrapper');
 }
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="primarybtn">Enable/Disable TreeGrid</button>
 <div id="TreeGridParent">

```

```

 <div id="TreeGrid"></div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

Change header text dynamically in EJ2 JavaScript Treegrid control

You can change the column [headerText](#) dynamically through an external button.

Follow the given steps to change the header text dynamically:

#### Step 1:

Get the column object corresponding to the field name by using the [getColumnByField](#) method. Then, change the header text value.

```

`ts
let column = treegrid.getColumnByField("taskName"); // Get column object.
column.headerText = 'Changed Text';
`

```

#### Step 2:

To reflect the changes in the treegrid header, invoke the [refreshColumns](#) method.

```

`ts
treegrid.refreshColumns();
`

```

#### INDEX.TS

```

import { TreeGrid, Page, Column } from '@syncfusion/ej2-treegrid';
import { Button } from '@syncfusion/ej2-buttons';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page);
let button: Button = new Button();
button.appendTo("#change-text");
let treegrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 90, textAlign: 'Right'
 },

```

```

 { field: 'taskName', headerText: 'Task Name', width: 180, textAlign:
 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90, textAlign:
 'Right', type: 'date', format: 'yMd'},
 { field: 'duration', headerText: 'Duration', width: 80, textAlign:
 'Right' }
],
});
treegrid.appendTo('#TreeGrid');
document.getElementById("change-text").addEventListener("click", () => {
 let column = treegrid.getColumnByField("taskName"); // get the JSON object
of the column corresponding to the field name.
 column.headerText = "Changed Text"; // assign a new header text to the
column.
 treegrid.refreshColumns();
});

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <button id="change-text"> Change Header Text</button>
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Customize column styles in EJ2 JavaScript Treegrid control

You can customise the appearance of the header and content of a particular column using the [customAttributes](#) property.

To customize the Tree Grid column, follow the given steps:

##### Step 1:

Create a CSS class with custom style to override the default style for rowcell and header cell.

```

.e-treegrid .e-rowcell.customcss{
background-color: #ecedee;
color: 'red';
font-family: 'Bell MT';
font-size: 20px;
}
.e-treegrid .e-headercell.customcss{
background-color: #2382c3;
color: white;
font-family: 'Bell MT';
font-size: 20px;
}

```



**Step 2:**

Add the custom CSS class to the specified column by using the [customAttributes](#) property.

```
`ts
```

```
{ field: 'taskName', headerText: 'Task Name', customAttributes: {class: 'customcss'}, width: 100 },
```

**INDEX.TS**

```
import { TreeGrid } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treegrid: TreeGrid = new TreeGrid(
 {
 dataSource: sampleData,
 childMapping: 'subtasks',
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width:
200,customAttributes: { class: 'customcss' }, textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
]
 });
treegrid.appendTo('#TreeGrid');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Custom tool tip for columns in EJ2 JavaScript Treegrid control

You can show the tooltip content in Tree Grid using `[queryCellInfo]` event of Tree Grid.

In the below demo, hover over the cell shows the tooltip content .

#### INDEX.TS

```

import { TreeGrid, Page, QueryCellInfoEventArgs } from '@syncfusion/ej2-
treegrid';
import { Tooltip } from '@syncfusion/ej2-popups';
import { sampleData } from './datasource.ts';
TreeGrid.Inject(Page);
let treegrid: TreeGrid = new TreeGrid(
{
 dataSource: sampleData,
 childMapping: 'subtasks',
 allowPaging: true,

```

```

 queryCellInfo: tooltip,
 pageSettings: {pageSize: 10},
 treeColumnIndex: 1,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70,
textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', width: 200,
textAlign: 'Left' },
 { field: 'startDate', headerText: 'Start Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'endDate', headerText: 'End Date', width: 90,
textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
textAlign: 'Right' },
 { field: 'priority', headerText: 'Priority', width: 90 }
]
 });
 treegrid.appendTo('#TreeGrid');
 function tooltip(args: QueryCellInfoEventArgs): void { // event triggers
on every cell render.
 let tooltip: Tooltip = new Tooltip({
 content: args.data[args.column.field].toString() // add Essential
JS2 tooltip for every cell.
 }, <HTMLElement>args.cell);
 }

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

Change orientation of header text in EJ2 JavaScript Treegrid control

You can change the orientation of the header text by using the [customAttributes](#) property.

Ensure the following steps:

#### Step 1:

Create a CSS class with orientation style for the treegrid header cell.

```

,

.orientationcss .e-headercelldiv {
transform: rotate(90deg);
}
,

```

#### Step 2:

Add the custom CSS class to a particular column by using the [customAttributes](#) property.

```

`ts

```

```
{ field: 'taskName', headerText: 'Task Name', textAlign: 'Center', customAttributes: {class: 'orientationcss'}, width: 80 }
```

```
,
```

### Step 3:

Resize the header cell height by using the following code.

```
`ts
```

```
function setHeaderHeight(args) {
```

```
let textWidth: number = document.querySelector(".orientationcss > div").scrollWidth; //Obtain the width of the headerText content.
```

```
let headerCell: NodeList = document.querySelectorAll(".e-headercell");
```

```
for(let i: number = 0; i < headerCell.length; i++) {
```

```
(<HTMLElement>headerCell.item(i)).style.height = textWidth + 'px'; //Assign the obtained textWidth as the height of the headerCell.
```

```
}
```

```
}
```

```
,
```

### INDEX.TS

```
import { TreeGrid, ActionEventArgs } from '@syncfusion/ej2-treegrid';
import { sampleData } from './datasource.ts';
let treegrid: TreeGrid = new TreeGrid(
{
 dataSource: sampleData,
 childMapping: "subtasks",
 treeColumnIndex: 1,
 height: 100,
 created: setHeaderHeight,
 columns: [
 { field: 'taskID', headerText: 'Task ID', width: 70,
 textAlign: 'Right' },
 { field: 'taskName', headerText: 'Task Name', textAlign:
 'Center', customAttributes: {class: 'orientationcss'}, width: 80 },
 { field: 'startDate', headerText: 'Start Date', width: 90,
 textAlign: 'Right', type: 'date', format: 'yMd' },
 { field: 'duration', headerText: 'Duration', width: 80,
 textAlign: 'Right' }
]
});
treegrid.appendTo('#TreeGrid');
function setHeaderHeight(args: ActionEventArgs): void {
let textWidth: number = document.querySelector(".orientationcss >
div").scrollWidth; // obtain the width of the headerText content.
let headerCell: NodeList = document.querySelectorAll(".e-headercell");
for (let i: number = 0; i < headerCell.length; i++) {
 (<HTMLElement>headerCell.item(i)).style.height = textWidth + 'px';
 // assign the obtained textWidth as the height of the headerCell.
}
}
```

```
}
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
```

```
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Render other component in column in EJ2 JavaScript Treegrid control

You can render any components in a Tree Grid column using the template property.

Initialize the column template for your custom component. The template property renders the custom component.

In the following sample, the DropDownList is rendered in the **Priority** column.

#### INDEX.TS

```
import { TreeGrid, Edit, Toolbar } from "@syncfusion/ej2-treegrid";
import { projectData } from '../datasource.ts';
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { QueryCellInfoEventArgs } from "@syncfusion/ej2-grids";
let dropData = ["Normal", "Low", "High", "Critical", "Breaker"];
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 queryCellInfo: dropdown,
 height: 273,
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right", width: 100,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width: 90 },
 { headerText: "Priority", width: 90, template: `<input type="text"
 tabindex="1" id='ddlelement' />` }
]
});
treegrid.appendTo("#TreeGrid");
function dropdown(args: QueryCellInfoEventArgs) {
 let ele: HTMLSelectElement = args.cell.querySelector("#ddlelement");
 let drop: DropDownList = new DropDownList({
 dataSource: dropData,
 value: args.data["Priority"],
 popupHeight: 150,
 popupWidth: 150,
 change: valueChange
 });
}
```

```

 drop.appendTo(ele);
}
function valueChange(args) {
 /** Event will trigger when you have change the value in dropdown column
 */
 alert(args.value);
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>

```



```

 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Customize the icon for column menu in EJ2 JavaScript Treegrid control

You can customize the column menu icon by overriding the default Tree Grid class **.e-icons.e-columnmenu** with a custom property **content** as mentioned below,

```

,
.e-treegrid .e-columnheader .e-icons.e-columnmenu::before {
content: "\e903";
}
,

```

In the below sample, Tree Grid is rendered with a customized column menu icon.

### INDEX.TS

```

import { TreeGrid, ColumnMenu } from "@syncfusion/ej2-treegrid";
import { projectData } from '../datasource.ts';
TreeGrid.Inject(ColumnMenu);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 315,
 showColumnMenu: true,
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70 },
],
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width:
90 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});

```

```
treegrid.appendTo("#TreeGrid");
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Customize the edit dialog in EJ2 JavaScript Treegrid control

You can customize the appearance of the edit dialog in the [actionComplete](#) event based on **requestType** as **beginEdit** or **add**.

In the below example, we have changed the dialog's header text for editing and adding records.

### INDEX.TS

```

import { TreeGrid, Toolbar, Edit } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { DialogEditEventArgs } from "@syncfusion/ej2-grids";
import { Dialog } from "@syncfusion/ej2-popups";
TreeGrid.Inject(Toolbar, Edit);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 250,
 editSettings: {
 allowAdding: true,
 allowEditing: true,
 allowDeleting: true,
 mode: "Dialog"
 },
 toolbar: ["Add", "Edit", "Delete", "Update", "Cancel"],
 actionComplete: actionComplete,
 columns: [
 { field: "TaskID", headerText: "Task ID", isPrimaryKey: true, textAlign: "Right", width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right", width: 90, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" }, edit: { params: { format: "y/M/d" } } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width: 90, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" }, edit: { params: { format: "y/M/d" } } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width: 90 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
function actionComplete(args: DialogEditEventArgs) {
 if (args.requestType === "beginEdit" || args.requestType === "add") {
 const dialog = args.dialog as Dialog;

```

```

const TaskName = "TaskName";
dialog.height = 400;
// change the header of the dialog
dialog.header =
 args.requestType === "beginEdit"
 ? "Record of " + args.rowData[TaskName]
 : "New Customer";
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

<div id="container">
 <div id="TreeGrid"></div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

Cascading drop down list with treegrid editing in EJ2 JavaScript Treegrid control

You can achieve the Cascading DropDownList with Tree Grid Editing by using the Cell Edit Template feature.

In the below demo, Cascading DropDownList rendered for **Priority** and **Duration** column.

#### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from "@syncfusion/ej2-treegrid";
import { projectData } from '../datasource.ts';
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { DataManager, Query } from "@syncfusion/ej2-data";
let priorityData: { [key: string]: Object }[] = [
 { priorityName: "Normal", priorityId: "1" },
 { priorityName: "High", priorityId: "2" }
];
let durationData: { [key: string]: Object }[] = [
 { durationValue: 2, priorityId: "1", durationId: 2 },
 { durationValue: 3, priorityId: "1", durationId: 3 },
 { durationValue: 4, priorityId: "1", durationId: 4 },
 { durationValue: 11, priorityId: "2", durationId: 11 },
 { durationValue: 15, priorityId: "2", durationId: 15 },
 { durationValue: 20, priorityId: "2", durationId: 20 }
];
let priorityElem: HTMLElement;
let priorityObj: DropDownList;
let durationElem: HTMLElement;
let durationObj: DropDownList;
TreeGrid.Inject(Edit, Toolbar);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 273,
 toolbar: ["Add", "Edit", "Delete", "Update", "Cancel"],
 editSettings: {
 allowEditing: true,
 allowAdding: true,
 allowDeleting: true,

```

```

 mode: "Row"
 },
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right",
isPrimaryKey: true, width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 100, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
90, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" }
 },
 { field: "Priority", headerText: "Priority", width: 90, editType:
"dropdownedit",
 edit: {
 create: () => {
 priorityElem = document.createElement("input");
 return priorityElem;
 },
 read: () => {
 return priorityObj.text;
 },
 destroy: () => {
 priorityObj.destroy();
 },
 write: () => {
 priorityObj = new DropDownList({
 dataSource: new DataManager(priorityData),
 fields: { value: "priorityId", text: "priorityName" },
 change: () => {
 durationObj.enabled = true;
 let tempQuery: Query = new Query().where(
 "priorityId",
 "equal",
 priorityObj.value
);
 durationObj.query = tempQuery;
 durationObj.text = null;
 durationObj.dataBind();
 },
 placeholder: "Select a priority",
 floatLabelType: "Never"
 });
 priorityObj.appendTo(priorityElem);
 }
 }
 },
 { field: "Duration", headerText: "Duration", textAlign: "Right",
editType: "dropdownedit", width: 90,
 edit: {
 create: () => {
 durationElem = document.createElement("input");
 return durationElem;
 },
 read: () => {
 return durationObj.text;
 },

```

```

 },
 destroy: () => {
 durationObj.destroy();
 },
 write: () => {
 durationObj = new DropDownList({
 dataSource: new DataManager(durationData),
 fields: { value: "durationId", text: "durationValue" },
 enabled: false,
 placeholder: "Select a duration",
 floatLabelType: "Never"
 });
 durationObj.appendTo(durationElem);
 }
 },
 { field: "Progress", headerText: "Progress", textAlign: "Right", width:
90 }
]
});
treegrid.appendTo("#TreeGrid");

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

Provide custom data source and enabling filtering to drop down list in EJ2 JavaScript Treegrid control

You can provide data source to the DropDownList by using the **params** of [columns.edit](#) property.

While setting new data source using edit params, you must specify a new **query** property for the DropDownList as follows,

```

`ts
edit: {
params: {
actionComplete: () => false,
allowFiltering: true,
dataSource: new DataManager(priorityData),
fields: { text: "priorityName", value: "priorityName" },
query: new Query()
}
}
`

```



You can also enable filtering for the DropDownList by passing the [allowFiltering](#) as **true** to the edit params.

In the below demo, DropDownList is rendered with custom [dataSource](#) for the *Priority* column and enabled filtering to search DropDownList items.

### INDEX.TS

```
import { TreeGrid, Edit, Toolbar } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { Query, DataManager } from "@syncfusion/ej2-data";
import { IEditCell } from "@syncfusion/ej2-grids";
TreeGrid.Inject(Edit, Toolbar);
let priorityData: object[] = [
 { priorityName: "Normal", priorityId: "1" },
 { priorityName: "High", priorityId: "2" },
 { priorityName: "Low", priorityId: "3" },
 { priorityName: "Critical", priorityId: "4" },
 { priorityName: "Breaker", priorityId: "5" }
];
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 editSettings: {
 allowEditing: true,
 allowAdding: true,
 allowDeleting: true,
 mode: "Row"
 },
 toolbar: ["Add", "Edit", "Delete", "Update", "Cancel"],
 columns: [
 { field: "TaskID", headerText: "Task ID", isPrimaryKey: true, textAlign: "Right", width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right", width: 100, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width: 100, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width: 90 },
 { field: "Priority", headerText: "Priority", width: 90, editType: "dropdownedit",
 edit: {
 params: {
 actionComplete: () => false,
 allowFiltering: true,
 dataSource: new DataManager(priorityData),
 fields: { text: "priorityName", value: "priorityName" },
 query: new Query()
 }
 }
 }
]
});
```

```
});
treegrid.appendTo("#TreeGrid");
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Use wizard like dialog editing in EJ2 JavaScript Treegrid control

Wizard helps you create intuitive step-by-step forms to fill. You can achieve the wizard like editing by using the dialog template feature. It support your own editing template by defining [editSettings.mode](#) as **Dialog** and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

The following example demonstrate the wizard like editing in the Tree Grid with the obtrusive Validation.

#### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { DataUtil } from "@syncfusion/ej2-data";
import { DialogEditEventArgs } from "@syncfusion/ej2-grids";
import { NumericTextBox } from "@syncfusion/ej2-inputs";
TreeGrid.Inject(Edit, Toolbar);
let PriorityData: {}[] = DataUtil.distinct(projectData, "Priority", true);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 273,
 toolbar: ["Add", "Edit", "Delete", "Update", "Cancel"],
 editSettings: {
 allowEditing: true,
 allowAdding: true,
 allowDeleting: true,
 mode: "Dialog",
 template: "#dialogtemplate"
 },
 actionComplete: actionComplete,
 columns: [
 { field: "TaskID", headerText: "Task ID", isPrimaryKey: true, textAlign: "Right", width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right", width: 90, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" }, edit: { params: { format: "y/M/d" } } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width: 90, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" }, edit: { params: { format: "y/M/d" } } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width: 90 },
],

```

```

 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
function actionComplete(args) {
 if (args.requestType === "beginEdit" || args.requestType === "add") {
 new DropDownList(
 {
 value: args.rowData.Priority,
 popupHeight: "300px",
 floatLabelType: "Always",
 dataSource: PriorityData,
 fields: { text: "Priority", value: "Priority" },
 placeholder: "Priority"
 },
 args.form.elements.namedItem("Priority") as HTMLInputElement
);
 new NumericTextBox(
 { value: args.rowData.Duration, placeholder: "Duration" },
 args.form.elements.namedItem("Duration")
);
 // Set initail Focus
 if (args.requestType === "beginEdit") {
 (args.form.elements.namedItem("TaskName") as
HTMLInputElement).focus();
 }
 initializeWizard();
 }
}
function initializeWizard() {
 let currentTab = 0;
 document.getElementById("nextBtn").onclick = function() {
 if (validate()) {
 if (this.innerHTML !== "SUBMIT") {
 currentTab++;
 nextpre(currentTab);
 } else {
 treegrid.endEdit();
 }
 }
 };
 function validate(tab) {
 let valid: boolean = true;
 [].slice
 .call(
 document.getElementById("tab" +
currentTab).querySelectorAll("[name]")
)
 .forEach(element => {
 element.form.ej2_instances[0].validate(element.name);
 if (element.getAttribute("aria-invalid") === "true") {
 valid = false;
 }
 });
 if (!valid) {
 return false;
 }
 }
}

```

```

 return true;
 }
 document.getElementById("prevBtn").onclick = function() {
 if (validate()) {
 currentTab--;
 nextpre(currentTab);
 }
 };
}
function nextpre(current) {
 let tabs: HTMLElement[] = [].slice.call(
 document.getElementsByClassName("tab")
);
 tabs.forEach(element => (element.style.display = "none"));
 tabs[current].style.display = "";
 if (current) {
 document.getElementById("prevBtn").style.display = "";
 document.getElementById("nextBtn").innerHTML = "SUBMIT";
 } else {
 document.getElementById("prevBtn").style.display = "none";
 document.getElementById("nextBtn").innerHTML = "NEXT";
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script id="dialogtemplate" type="text/x-template">
 <div>
 <div id="tab0" class='tab'>
 <div class="form-row">
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input id="TaskID" required name="TaskID"
type="text" value=${if(isAdd)} '' ${else} ${TaskID} ${if} ${if(isAdd)} ''
${else} disabled ${if} />

 <label class="e-float-text e-label-top"
for="TaskID">TaskID</label>
 </div>
 </div>
 </div>
 <div class="form-row">
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input id="TaskName" required name="TaskName"
type="text" value=${if(isAdd)} '' ${else} ${TaskName} ${if} />

 <label class="e-float-text e-label-top"
for="TaskName">TaskName</label>
 </div>
 </div>
 </div>
 </div>
 <div id=tab1 style="display: none" class='tab'>
 <div class="form-row">
 <div class="form-group col-md-6">
 <input type="text" name="Duration" id="Duration"
value=${if(isAdd)} '' ${else} ${Duration} ${if} />
 </div>
 </div>
 <div class="form-row">
 <div class="form-group col-md-6">
 <input type="text" name="Priority" id="Priority"
value=${if(isAdd)} '' ${else} ${Priority} ${if} />
 </div>
 </div>
 </div>
 </div>
 </script>

```

```

 </div>
 <div id='footer'>
 <button id="prevBtn" class="e-info e-btn" type="button"
style="display: none; float: left">Previous</button>

 <button id="nextBtn" class="e-info e-btn" type="button"
style="float: right">Next</button>
 </div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

Using tab inside the dialog editing in EJ2 JavaScript Treegrid control

You can use [tab](#) component inside dialog edit UI using dialog template feature. The dialog template feature can be enabled by defining [editSettings.mode](#) as `Dialog` and [editSetting.template](#) as SCRIPT element ID or HTML string which holds the template.

The following example demonstrate the usage of tab control inside the dialog template.

#### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { DropDownList } from "@syncfusion/ej2-dropdowns";
import { DataUtil } from "@syncfusion/ej2-data";
import { DialogEditEventArgs } from "@syncfusion/ej2-grids";
import { NumericTextBox } from "@syncfusion/ej2-inputs";
import { Tab } from "@syncfusion/ej2-navigations";
TreeGrid.Inject(Edit, Toolbar);
let PriorityData: {}[] = DataUtil.distinct(projectData, "Priority", true);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 273,
 toolbar: ["Add", "Edit", "Delete", "Update", "Cancel"],
 editSettings: {
 allowEditing: true,
 allowAdding: true,
 allowDeleting: true,
 mode: "Dialog",
 template: "#dialogtemplate"
 },
 actionComplete: actionComplete,

```

```

columns: [
 { field: "TaskID", headerText: "Task ID", isPrimaryKey: true, textAlign:
"Right", width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 90, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" }, edit: { params: { format:
"y/M/d" } } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
90, editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" }, edit: { params: { format:
"y/M/d" } } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width:
90 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");function actionComplete(args:
DialogEditEventArgs) {
 if (args.requestType === "beginEdit" || args.requestType === "add") {
 let tabObj: Tab = new Tab({
 showCloseButton: false,
 selecting: e => {
 if (e.isSwiped) {
 e.cancel = true;
 }
 if (e.selectingIndex === 1) {
 e.cancel = !validate(1);
 }
 },
 items: [
 { header: { text: "Details" }, content: "#tab1" },
 { header: { text: "Verify" }, content: "#tab2" }
]
 });
 tabObj.appendTo("#edittab");
 new DropDownList(
 {
 value: args.rowData.Priority,
 popupHeight: "300px",
 floatLabelType: "Always",
 dataSource: PriorityData,
 fields: { text: "Priority", value: "Priority" },
 placeholder: "Priority"
 },
 args.form.elements.namedItem("Priority") as HTMLInputElement
);
 new NumericTextBox(
 { value: args.rowData.Duration, placeholder: "Duration" },
 args.form.elements.namedItem("Duration")
);
 // Set initail Focus
 if (args.requestType === "beginEdit") {
 (args.form.elements.namedItem("TaskName") as
HTMLInputElement).focus();
 }
 document.getElementById("next").onclick = () => {

```



```

 moveNext();
 };
 function validate(tab) {
 let valid: boolean = true;
 [].slice
 .call(document.getElementById("tab" +
tab).querySelectorAll("[name]"))
 .forEach(element => {
 element.form.ej2_instances[0].validate(element.name);
 if (element.getAttribute("aria-invalid") === "true") {
 valid = false;
 }
 });
 if (!valid) {
 return false;
 }
 return true;
 }
 function moveNext() {
 if (validate(1)) {
 tabObj.select(1);
 }
 }
 document.getElementById("submit").onclick = () => {
 if (validate(2)) {
 treegrid.endEdit();
 }
 };
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script id="dialogtemplate" type="text/x-template">
 <div>
 <div id="edittab"></div>
 <div id="tab1">
 <div class="form-row" >
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input id="TaskID" name="TaskID" type="text"
value=${if(isAdd)} '' ${else} ${TaskID} ${/if} ${if(isAdd)} '' ${else}
disabled ${/if} />

 <label class="e-float-text e-label-top"
for="TaskID">Task ID</label>
 </div>
 </div>
 </div>
 <div class="form-row" >
 <div class="form-group col-md-6">
 <div class="e-float-input e-control-wrapper">
 <input id="TaskName" name="TaskName" type="text"
value=${if(isAdd)} '' ${else} ${TaskName} ${/if} />

 <label class="e-float-text e-label-top"
for="CustomerID">Task Name</label>
 </div>
 </div>
 </div>
 <button id='next' class='e-info e-btn' style="float: right"
type='button'> next</button>
 </div>
 <div id="tab2" style='display: none'>
 <div class="form-row" >

```

```

 <div class="form-group col-md-6">
 <input type="text" name="Duration" id="Duration"
value=${if(isAdd)} '' ${else} ${Duration} ${if} />
 </div>
 </div>
 <div class="form-row">
 <div class="form-group col-md-6">
 <input type="text" name="Priority" id="Priority"
value=${if(isAdd)} '' ${else} ${Priority} ${if} />
 </div>
 <button id='submit' class='e-info e-btn' style="float:
right" type='button'> submit</button>
 </div>
</div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Restrict decimal points while treegrid editing in EJ2 JavaScript Treegrid control

By default, the number of decimal places will be restricted to two in the NumericTextBox while editing the numeric column. We can restrict to type the decimal points in a NumericTextBox by using the **validateDecimalOnType** and **decimals** properties of NumericTextBox.

In the below demo, while editing the row we have restricted to type the decimal point value in the NumericTextBox of **Price** column.

### INDEX.TS

```

import { TreeGrid, Edit, Toolbar } from "@syncfusion/ej2-treegrid";
import { stackedData } from './datasource.ts';
TreeGrid.Inject(Edit, Toolbar);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: stackedData,
 childMapping: "subtasks",
 treeColumnIndex: 1,
 height: 265,
 editSettings: { allowEditing: true, allowAdding: true, allowDeleting: true },
 toolbar: ["Add", "Delete", "Update", "Cancel"],
 columns: [
 { field: "orderID", headerText: "Order ID", textAlign: "Right",
isPrimaryKey: true, width: 70 },
 { field: "orderName", headerText: "Order Name", width: 100 },
 { field: "orderDate", headerText: "Order Date", textAlign: "Right",
width: 100, editType: "datepickeredit",

```

```

 format: { skeleton: "yMd", type: "date" } },
 { field: "shippedDate", headerText: "Shipped Date", textAlign: "Right",
width: 100,
 editType: "datepickeredit", format: { skeleton: "yMd", type: "date" }
 },
 { field: "shipMentCategory", headerText: "Shipment Category", textAlign:
"Right", width: 100 },
 { field: "units", headerText: "Units", width: 90, editType:
"numericedit" },
 { field: "price", headerText: "Price", width: 90, editType:
"numericedit",
 edit: {
 params: {
 validateDecimalOnType: true,
 decimals: 0,
 format: "N"
 }
 }
 }
]
});
treegrid.appendTo("#TreeGrid");

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Exporting filtered data in EJ2 JavaScript Treegrid control

You can export the filtered data by defining the resulted data in [PdfExportProperties.dataSource](#) before export.

In the below Pdf exporting demo, We have gotten the filtered data from the filteredResult of Tree Grid filterModule and then defines the resulted data in [PdfExportProperties.dataSource](#) and pass it to [pdfExport](#) method.

### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport, Filter } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { ClickEventArgs } from "@syncfusion/ej2-navigations";
TreeGrid.Inject(Page, Filter, Toolbar, PdfExport);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 allowPaging: true,
 allowFiltering: true,
 pageSettings: { pageSize: 5, pageCount: 5 },
 allowExcelExport: true,
 allowPdfExport: true,
 toolbar: ["PdfExport"],

```

```

toolbarClick: toolbarClick,
columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70,
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 100,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
100,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width:
90 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
function toolbarClick(args: ClickEventArgs) {
 if (treegrid && args.item.text === "PDF Export") {
 let pdfdata;
 pdfdata = treegrid.filterModule.filteredResult;
 const exportProperties = {
 dataSource: pdfdata
 };
 if (treegrid) {
 treegrid.pdfExport(exportProperties);
 }
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Exporting selected data in EJ2 JavaScript Treegrid control

You can export the selected records data by passing it to [PdfExportProperties.dataSource](#) or [ExcelExportProperties.dataSource](#) property in the [toolbarClick](#) event.

In the below exporting demo, we can get the selected records using [getSelectedRecords](#) method and pass the selected data to [pdfExport](#) or [excelExport](#) methods using respective export properties..

#### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport, ExcelExport } from
"@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { ClickEventArgs } from "@syncfusion/ej2-navigations";
TreeGrid.Inject(Page, ExcelExport, Toolbar, PdfExport);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 allowPaging: true,
 pageSettings: { pageSize: 5, pageCount: 5 },

```

```

selectionSettings: { type: "Multiple" },
allowExcelExport: true,
allowPdfExport: true,
toolbar: ["PdfExport", "ExcelExport"],
toolbarClick: toolbarClick,
columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70
},
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 100,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
100,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width:
90 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
function toolbarClick(args: ClickEventArgs) {
 if (treegrid && args.item.text === "PDF Export") {
 const selectedRecords = treegrid.getSelectedRecords();
 const exportProperties = {
 dataSource: selectedRecords
 };
 treegrid.pdfExport(exportProperties);
 } else if (treegrid && args.item.text === "Excel Export") {
 const selectedRecords = treegrid.getSelectedRecords();
 const exportProperties = {
 dataSource: selectedRecords
 };
 treegrid.excelExport(exportProperties);
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```



```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Show spinner while exporting in EJ2 JavaScript Treegrid control

You can show/ hide spinner component while exporting the Tree Grid using [showSpinner/ hideSpinner](#) methods. You can use [toolbarClick](#) event to show spinner before exporting and hide a spinner in the [pdfExportComplete](#) or [excelExportComplete](#) event after the exporting.

In the [toolbarClick](#) event, based on the parameter **args.item.text** as **PDF Export** or **Excel Export** we can call the [showSpinner](#) method from Tree Grid instance.

In the [pdfExportComplete](#) or [excelExportComplete](#) event, We can call the [hideSpinner](#) method.

In the below demo, we have rendered the default spinner component when exporting the Tree Grid.

#### INDEX.TS

```

import { TreeGrid, Page, Toolbar, PdfExport, ExcelExport } from
"@syncfusion/ej2-treegrid";
import { projectData } from '../datasource.ts';
import { ClickEventArgs } from "@syncfusion/ej2-navigations";
import { Query } from "@syncfusion/ej2-data";
TreeGrid.Inject(Page, ExcelExport, Toolbar, PdfExport);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 allowPaging: true,
 pageSettings: { pageSize: 5, pageCount: 5 },
 allowExcelExport: true,
 allowPdfExport: true,
 pdfExportComplete: pdfExportComplete,
 excelExportComplete: excelExportComplete,
 toolbar: ["PdfExport", "ExcelExport"],
 toolbarClick: toolbarClick,
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70
 },
 { field: "TaskName", headerText: "Task Name", width: 150 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width:
80 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
function toolbarClick(args: ClickEventArgs) {
 if (treegrid && args.item.text === "PDF Export") {
 treegrid.showSpinner();
 treegrid.pdfExport();
 } else if (treegrid && args.item.text === "Excel Export") {
 treegrid.showSpinner();
 treegrid.excelExport();
 }
}
function pdfExportComplete(args) {
 if (treegrid) {
 treegrid.hideSpinner();
 }
}
function excelExportComplete(args) {
 if (treegrid) {
 treegrid.hideSpinner();
 }
}
}

```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Customize pager drop down in EJ2 JavaScript Treegrid control

To customize default values of pager dropdown, you need to define [pageSizes](#) as array of strings.

#### INDEX.TS

```
import { TreeGrid, Page } from "@syncfusion/ej2-treegrid";
import { projectData } from '../datasource.ts';
TreeGrid.Inject(Page);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 260,
 allowPaging: true,
 pageSettings: { pageSizes: ["5", "10", "All"] },
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 200 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right", width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width: 90 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Add parameter for filtering in EJ2 JavaScript Treegrid control

You can customize the default settings of the components which are used in Menu filter by using params of filter property in column definition.

In the below sample, TaskID and Duration Columns are numeric columns, while opening the filter dialog you can see that NumericTextBox with spin button is displayed to change/set the filter value. Now using the params option we hide the spin button in NumericTextBox for TaskID Column.

#### INDEX.TS

```

import { TreeGrid, Filter } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
TreeGrid.Inject(Filter);

```

```

let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 273,
 allowFiltering: true,
 filterSettings: { type: "Menu" },
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70,
 filter: { params: { showSpinButton: false } } },
 { field: "TaskName", headerText: "Task Name", width: 100 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 90, format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
90, format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width:
90 },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Tree Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Tree Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <!-- Syncfusion Essential JS 2 Styles -->
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>

 <div class="datalink">Source:
 <a
href="https://en.wikipedia.org/wiki/List_of_Android_smartphones"
target="_blank">Wikipedia: List of Android smartphones
 </div>
 </div>
 <script>
var ele = document.getElementById('container');

```

```

if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

Select treegrid rows based on certain condition in EJ2 JavaScript Treegrid control

You can select the specific row in the Tree Grid based on a certain condition by using the [selectRows](#) method in the [dataBound](#) event of Tree Grid.

In the below demo, we have selected the Tree Grid rows only when *Duration* column value greater than 4.

#### INDEX.TS

```

import { TreeGrid, Page } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { RowDataBoundEventArgs } from "@syncfusion/ej2-grids";
import { getValue } from "@syncfusion/ej2-base";
TreeGrid.Inject(Page);
let selIndex = [];
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 269,
 allowPaging: true,
 selectionSettings: { type: "Multiple" },
 dataBound: dataBound,
 rowDataBound: rowDataBound,
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70 },
 { field: "TaskName", headerText: "Task Name", width: 150 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right", width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width: 80 },
 { field: "Progress", headerText: "Progress", width: 80, textAlign: "Right" },
 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
function dataBound(args) {
 if (this && selIndex.length) {

```

```

 this.selectRows(selIndex);
 selIndex = [];
 }
}
function rowDataBound(args: RowDataBoundEventArgs) {
 if (getValue("Duration", args.data as object) > 4) {
 selIndex.push(
 parseInt(
 (args.row as HTMLTableRowElement).getAttribute(
 "aria-rowindex"
) as string,
 0
)
);
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

```



```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Get row cell index in EJ2 JavaScript Treegrid control

You can get the specific row and cell index of the Tree Grid by using [rowSelected](#) event of the treegrid. Here, we can get the row and cell index by using *aria-rowindex* (get row Index from *tr* element) and *aria-colindex* (column index from *td* element) attribute.

#### INDEX.TS

```

import { TreeGrid } from "@syncfusion/ej2-treegrid";
import { projectData } from './datasource.ts';
import { RowSelectEventArgs } from "@syncfusion/ej2-grids";
let treegrid: TreeGrid = new TreeGrid({
 dataSource: projectData,
 idMapping: "TaskID",
 parentIdMapping: "parentID",
 treeColumnIndex: 1,
 height: 267,
 rowSelected: rowSelected,
 columns: [
 { field: "TaskID", headerText: "Task ID", textAlign: "Right", width: 70
 },
 { field: "TaskName", headerText: "Task Name", width: 150 },
 { field: "StartDate", headerText: "Start Date", textAlign: "Right",
width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "EndDate", headerText: "End Date", textAlign: "Right", width:
90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "Duration", headerText: "Duration", textAlign: "Right", width:
80 },
 { field: "Progress", headerText: "Progress", width: 80, textAlign:
"Right" },
]

```

```

 { field: "Priority", headerText: "Priority", width: 90 }
]
});
treegrid.appendTo("#TreeGrid");
function rowSelected(args: RowSelectEventArgs) {
 alert("row index: " + " " + (args.row as
HTMLTableRowElement).getAttribute("aria-rowindex"));
 alert("column index: " + " " +
args.target.closest("td").getAttribute("aria-colindex"));
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="es5-datasource.js" type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

### Display foreign key values in treegrid in EJ2 JavaScript Treegrid control

Since Tree Grid Databinding concept is of hierarchy relationship, we do not provide in-built support for foreignKey datasource.

To display the foreignKey value at initial rendering, we can use the [queryCellInfo](#) event of the Tree Grid and also by using the [editType](#) and [columns.edit](#) properties of Tree Grid Column, we can render Dropdownlist with external or foreign dataSource.

### INDEX.TS

```

import { TreeGrid, Edit, Toolbar, ITreeData } from "@syncfusion/ej2-treegrid";
import { foreignKeyData, dropData } from '../datasource.ts';
import { DataManager, Query } from "@syncfusion/ej2-data";
import { QueryCellInfoEventArgs, Column, IEditCell } from "@syncfusion/ej2-grids";
TreeGrid.Inject(Edit, Toolbar);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: foreignKeyData,
 childMapping: "Children",
 treeColumnIndex: 1,
 height: 269,
 editSettings: {
 allowEditing: true,
 allowAdding: true,
 allowDeleting: true,
 mode: "Cell"
 },
 toolbar: ["Add", "Delete", "Update", "Cancel"],
 queryCellInfo: queryCellInfo,
 columns: [
 { field: "EmpID", headerText: "EmpID", textAlign: "Right", isPrimaryKey: true, width: 70 },
 { field: "Name", headerText: "Employee Name", width: 70 },
 { field: "DOB", headerText: "DOB", textAlign: "Right", width: 70,
 editType: "datepickeredit",
 format: { skeleton: "yMd", type: "date" } },
 { field: "EmployeeID", headerText: "EmployeeID", textAlign: "Right",
 editType: "dropdownedit",

```

```

 edit: {
 params: {
 dataSource: new DataManager(dropData),
 fields: { text: "EmployeeName", value: "EmployeeID" },
 query: new Query()
 }
 },
 width: 70 },
 { field: "Country", headerText: "Country", width: 90, textAlign: "Right"
 }
]
});
treegrid.appendTo("#TreeGrid");
function queryCellInfo(args: QueryCellInfoEventArgs) {
 if ((args.column as Column).field === "EmployeeID") {
 for (var i = 0; i < dropData.length; i++) {
 let data: Object = args.data as Object;
 if (data[(args.column as Column).field] === dropData[i]["EmployeeID"])
 {
 (args.cell as HTMLElement).innerText = dropData[i]["EmployeeName"];
 // assign the foreignkey field value to the innertext
 }
 }
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

#### Row cell customization in EJ2 JavaScript Treegrid control

In Tree Grid we can customize the row and cell using [queryCellInfo](#) and [rowDataBound](#) events of Tree Grid.

In the below demo, we customize and show the command buttons only for the parent rows using [queryCellInfo](#) and [rowDataBound](#) events of Tree Grid.

#### INDEX.TS

```

import { TreeGrid, CommandColumn, Column, ITreeData } from "@syncfusion/ej2-treegrid";
import { sampleData } from './datasource.ts';
import { QueryCellInfoEventArgs, RowDataBoundEventArgs } from "@syncfusion/ej2-grids";
TreeGrid.Inject(CommandColumn);
let treegrid: TreeGrid = new TreeGrid({
 dataSource: sampleData,
 childMapping: "subtasks",
 treeColumnIndex: 1,
 height: 280,
 rowDataBound: rowDataBound,
 queryCellInfo: queryCellInfo,
 columns: [

```

```

 { field: "taskID", headerText: "Task ID", isPrimaryKey: true, width: 80,
 textAlign: "Right" },
 { field: "taskName", headerText: "Task Name", width: 200 },
 { field: "startDate", headerText: "Start Date", textAlign: "Right",
 width: 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "endDate", headerText: "End Date", textAlign: "Right", width:
 90,
 format: { skeleton: "yMd", type: "date" } },
 { field: "duration", headerText: "Duration", textAlign: "Right", width:
 90 },
 { field: "progress", headerText: "Progress", width: 90 },
 { headerText: "Commands", width: 120,
 commands: [
 {
 buttonOption: {
 content: "Details",
 cssClass: "e-flat"
 }
 }
]
 }
]
});
treegrid.appendTo("#TreeGrid");
function rowDataBound(args: RowDataBoundEventArgs) {
 if (!(args.data as ITreeData).hasChildRecords) {
 (args.row as HTMLElement).style.backgroundColor = "green";
 }
}
function queryCellInfo(args: QueryCellInfoEventArgs) {
 if (!(args.data as ITreeData).hasChildRecords) {
 if ((args.cell as HTMLElement).classList.contains("e-unboundcell")) {
 (args.cell as HTMLElement).querySelector(
 ".e-unboundcelldiv"
) as HTMLElement).style.display = "none";
 }
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>EJ2 Grid</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Typescript Grid Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
treegrid/styles/material.css" rel="stylesheet">

```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
dropdowns/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
lists/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
calendars/styles/material.css" rel="stylesheet">

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
splitbuttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="es5-datasource.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="TreeGrid"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

You can refer to our [JavaScript Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our JavaScript Tree Grid example [JavaScript Tree Grid example](#) to know how to present and manipulate data.

## Treemap

### Data binding in EJ2 JavaScript Treemap control

The TreeMap control supports data binding using the dataSource property.

### Populate data

The `dataSource` property accepts collection values as input. For example, a list of objects can be provided as input. Data can be given as either flat or hierarchical collection to the `dataSource` property.

### Flat collection

The following code shows, how to bind a flat collection as data source to the TreeMap control.

#### INDEX.TS

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {
 labelPath: 'State'
 },
}, '#container');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
```



```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### *Hierarchical collection*

The following code shows, how to bind a hierarchical collection as data source to the TreeMap control.

#### **INDEX.TS**

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [{
 "Continent": [{
 "Name": "Africa",
 "Population": 1216130000,
 "States": [{
 "Name": "Eastern Africa",
 "Population": 410637987,
 "Region": [{
 "Name": "Ethiopia",
 "Population": 107534882
 }]
 },
 {
 "Name": "Middle Africa",
 "Population": 158562976,
 "Region": [{
 "Name": "Democratic, Republic
of the Congo",
 "Population": 84004989
 }]
 }
]
 },
 {
 "Continent": [{
 "Name": "Asia",
 "Population": 4436224000,
 "States": [{
 "Name": "Central Asia",
 "Population": 69787760,
 "Region": [{
 "Name": "Uzbekistan",
 "Population": 32364996
 }]
 },
 {
 "Name": "Eastern Asia",
 "Population": 1641908531,
 "Region": [{
 "Name": "China",
 "Population": 1415045928
 }]
 }
]
 }
]
}]
```

```

 },
 {
 "Continent": [{
 "Name": "North America",
 "Population": 579024000,
 "States": [{
 "Name": "Central America",
 "Population": 174988756,
 "Region": [{
 "Name": "Mexico",
 "Population": 130759074
 }]
 },
 {
 "Name": "Northern America",
 "Population": 358593810,
 "Region": [{
 "Name": "U.S.",
 "Population": 3267667480
 }]
 }
]
 }
],
{
 "Continent": [{
 "Name": "South America",
 "Population": 422535000,
 "States": [{
 "Name": "Brazil",
 "Population": 204519000
 }]
 }
],
{
 "Continent": [{
 "Name": "Europe",
 "Population": 738849000,
 "States": [{
 "Name": "Eastern Europe",
 "Population": 291953328,
 "Region": [{
 "Name": "Russia",
 "Population": 143964709
 }]
 },
 {
 "Name": "Northern Europe",
 "Population": 103642971,
 "Region": [{
 "Name": "United Kingdom",
 "Population": 66573504
 }]
 }
]
}
]
}

```

```

],
 weightValuePath: 'Population',
 leafItemSettings: {
 labelPath: 'Name',
 fill: '#0077b3',
 border: { color: 'black', width: 0.5 }
 },
 levels: [
 { groupPath: 'Continent', fill: '#004466', border: { color: 'black', width: 0.5 } },
 { groupPath: 'States', fill: '#0099e6', border: { color: 'black', width: 0.5 } },
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Layout in EJ2 JavaScript Treemap control

Determine the visual representation of nodes belonging to all the TreeMap levels using the `layoutType` property.

### Types of layout

The available layout types are,

- Squarified
- SliceAndDiceVertical

- SliceAndDiceHorizontal
- SliceAndDiceAuto

### Squarified

The **Squarified** layout displays the nested rectangles based on aspect ratio in the TreeMap. The rectangles will be split based on the height and width of the parent. The default rendering type of layout is **Squarified**.

### INDEX.TS

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP'
}, '#container');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

### *SliceAndDiceVertical*

The **SliceAndDiceVertical** layout creates rectangles with high aspect ratio and displays items in a vertically sorted order.

#### **INDEX.TS**

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 layoutType : 'SliceAndDiceVertical',
 weightValuePath: 'GDP'
}, '#container');
```

#### **INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

*SliceAndDiceHorizontal*

The **SliceAndDiceHorizontal** layout creates rectangles with high aspect ratio and displays items in a horizontally sorted order.

**INDEX.TS**

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 layoutType : 'SliceAndDiceHorizontal',
 weightValuePath: 'GDP'
}, '#container');
```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

### *SliceAndDiceAuto*

The **SliceAndDiceAuto** layout creates rectangles with high aspect ratio and display items sorted both horizontally and vertically.

#### **INDEX.TS**

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 layoutType : 'SliceAndDiceAuto',
 weightValuePath: 'GDP'
}, '#container');
```

#### **INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## Leaf item in EJ2 JavaScript Treemap control

A leaf item defines a visualized data element and does not contain child nodes but contains parent node if the levels are specified in the TreeMap.

### Leaf label

Label is represented by item name or value. Label will be appeared by specifying the `labelPath` property and customize the label style using the `labelStyle` property.

### INDEX.TS

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {
 labelPath: 'State',
 labelStyle: {
 color: '#000000'
 },
 border: {
 color: '#000000',
 width: 0.5
 }
 },
}, '#container');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>
```



```

 <div id="container">
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Label position and format

Positioning the leaf item label using the `labelPosition` property and the text format can be customized by specifying data source properties name in the `labelFormat` property.

### INDEX.TS

```

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {
 labelPath: 'State',
 labelPosition: 'TopCenter',
 labelFormat: '${State}
${GDP} Trillion
(${percentage}
 %)',
 },
}, '#container');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Label template and position

Specifies the template of leaf item label and position of the template to be customized using `labelTemplate` and `templatePosition` properties.

### INDEX.TS

```

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 { Sport: "Swimming", Gold: 16, GameImage: 'Swimming.svg', ItemHeight:
"180px", ItemWidth: '180px' },
 { Sport: "Athletics", Gold: 13, GameImage: 'Athletics.svg', ItemHeight:
"70px", ItemWidth: '70px' },
 { Sport: "Gymnastics", Gold: 4, GameImage: 'Gymnastics.svg', ItemHeight:
"80px", ItemWidth: '80px' },
 { Sport: "Cycling", Gold: 2, GameImage: 'Cycling.svg', ItemHeight:
"50px", ItemWidth: '50px' },
 { Sport: "Wrestling", Gold: 2, GameImage: 'Wrestling.svg', ItemHeight:
"60px", ItemWidth: '50px' },
 { Sport: "Basketball", Gold: 2, GameImage: 'Basketball.svg', ItemHeight:
"50px", ItemWidth: '50px' },
 { Sport: "Boxing", Gold: 1, GameImage: 'Boxing.svg', ItemHeight: "40px",
ItemWidth: '30px' },
 { Sport: "Tennis", Gold: 1, GameImage: 'Tennis.svg', ItemHeight: "40px",
ItemWidth: '40px' },
 { Sport: "Judo", Gold: 1, GameImage: 'Judo.svg', ItemHeight: "40px",
ItemWidth: '40px' },
 { Sport: "Rowing", Gold: 1, GameImage: 'Rowing.svg', ItemHeight: "40px",
ItemWidth: '40px' },
 { Sport: "Shooting", Gold: 1, GameImage: 'Shooting.svg', ItemHeight:
"40px", ItemWidth: '40px' },
 { Sport: "Triathlon", Gold: 1, GameImage: 'Triathlon.svg', ItemHeight:
"40px", ItemWidth: '40px' },
 { Sport: "Water polo", Gold: 1, GameImage: 'Water polo.svg', ItemHeight:
"40px", ItemWidth: '40px' }
],
 weightValuePath: 'Gold',
 leafItemSettings: {
 labelPath: 'Sport',
 fill: '#993399',

```

```

 templatePosition: 'Center'
 },
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Item gap

The **gap** property is used to separate an item from another item. Each item rectangle is split into equal space with specified gap.

## INDEX.TS

```

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {

```

```

 labelPath: 'State',
 gap:20
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Levels in EJ2 JavaScript Treemap control

Treemap supports a number of levels and each level is separated by using the `groupPath` property.

### Group path

The `groupPath` property is used to separate each level of the Treemap by specifying the property from the data source.

In the following example, three levels are added and each level is configured using the `groupPath` property.

## INDEX.TS

```

let jobData:Object[] = [
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },

```

```

 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
 EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
 EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
 EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
 EmployeesCount: 100 }];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
 "#009688"],
 dataSource: jobData,
 weightValuePath: 'EmployeesCount',
 levels: [
 { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobDescription', border: { color: 'black', width: 0.5 } }
],
 { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
 Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Group gap

The `groupGap` property is used to separate an item from each group or another item to differentiate the levels mentioned in the TreeMap.

### INDEX.TS

```

let jobData:Object[] = [
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
 EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
 EmployeesCount: 50 },

```

```

 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
 EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
 EmployeesCount: 100 }]];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
 "#009688"],
 dataSource: jobData,
 weightValuePath: 'EmployeesCount',
 levels: [
 { groupPath: 'Country', groupGap: 10, border: { color: 'black',
 width: 0.5 } },
 { groupPath: 'JobDescription', groupGap: 10, border: { color:
 'black', width: 0.5 } },
 { groupPath: 'JobGroup', groupGap: 10, border: { color: 'black',
 width: 0.5 } },
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Header format and Alignment

Customize header using the `headerFormat` property in which fields are mapping from the `dataSource` and align header using the `headerAlignment` property.

**INDEX.TS**

```

let jobData:Object[] = [
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
 EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
 EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
 EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
 EmployeesCount: 100 }];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
 "#009688"],
 dataSource: jobData,
 weightValuePath: 'EmployeesCount',
 levels: [
 { groupPath: 'Country', headerFormat: '${Country}-${EmployeesCount}',
 headerAlignment: 'Center', border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobDescription', headerFormat: '${JobDescription}-
 ${EmployeesCount}', headerAlignment: 'Far', border: { color: 'black', width:
 0.5 } },
 { groupPath: 'JobGroup', headerAlignment: 'Near',
 headerFormat: '${JobGroup}-${EmployeesCount}', border: { color:
 'black', width: 0.5 } },
]
}, '#container');

```



**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

**Header height and style**

Customize the font color, family, weight, opacity and size using the `headerStyle`. Based on the font settings, the header height is given using the `headerHeight` property in levels.

**INDEX.TS**

```

let jobData:Object[] = [
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },

```

```

 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }]];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
 dataSource: jobData,
 weightValuePath: 'EmployeesCount',
 levels: [
 { groupPath: 'Country', headerHeight:35, headerStyle:{ size:'15px'
}}, border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobDescription',headerHeight:45, headerStyle:{
size:'15px' }, border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobGroup',headerHeight:40, headerStyle:{ size:'15px'
}}, border: { color: 'black', width: 0.5 } },
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
</body>
</script>

```

```

var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Header template and position

The TreeMap header supports to customize header of each item using the `headerTemplate` property. It uses Essential JS2 Template engine to render the elements. You can position the template using the `templatePosition` property.

### INDEX.TS

```

let jobData:Object[] = [
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
 EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
 EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
 EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
 EmployeesCount: 100 }];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
 "#009688"],

```

```

 dataSource: jobData ,
 weightValuePath: 'EmployeesCount',
 levels: [
 { groupPath: 'Country',headerTemplate:'<div>{{Country}}</div>',
headerPosition:'Center', border: { color: 'black', width: 0.5 } },
 { groupPath:
'JobDescription',headerTemplate:'<div>{{JobDescription}}</div>',
headerPosition:'Center', border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobGroup',headerTemplate:'<div>{{JobGroup}}</div>',
headerPosition:'Far', border: { color: 'black', width: 0.5 } },
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Color mapping in EJ2 JavaScript Treemap control

Color mapping is used to customize the color for each group or item based on the specified types. The following options are available to customize the group and leaf items in the TreeMap.

### Range color mapping

Range color mapping is used to apply color to the items by giving specific ranges in the DataSource, and it should be specifying the data source properties to the `rangeColorValuePath`. The color mapping ranges to be specified in the `from` and `to` properties of the `colorMapping`.

## INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },

```

```

 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping: [
 {
 from: 500,
 to: 3000,
 color: 'orange'
 },
 {
 from: 3000,
 to: 5000,
 color: 'green'
 }
]
 }
}, '#container');
```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Equal color mapping

Equal color mapping is used to fill colors to each item by specifying equal value present in the data source, that can be specified in the `equalColorValuePath` property.

#### INDEX.TS

```
let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
 { Car:'RS7 Sportback', Brand:'Audi', count:523 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails,
 weightValuePath: 'count',
 equalColorValuePath: 'Brand',
 leafItemSettings: {
 labelPath: 'Car',
 colorMapping:[
 {
 value:'Ford',
 color:'green'
 },
 {
 value:'Audi',
 color:'red'
 },
 {
 value:'Maruti',
 color:'orange'
 }
]
 },
}, '#container');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Desaturation color mapping

Desaturation color mapping is used to apply colors to the items based on the `minOpacity` and `maxOpacity` properties in the `colorMapping`.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping:[
 {
 from:500,
 to:3000,
 minOpacity:0.2,
 maxOpacity:0.5,
 color:'orange'
 },
 {
 from:3000,
 to:5000,
 minOpacity:0.5,
 maxOpacity:0.8,
 color:'green'
 }
]
 }
}, '#container');

```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

**Palette color mapping**

The palette color mapping is used to fill the color to each group or leaf item by given colors in the **palette** property.

**INDEX.TS**

```

let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
 { Car:'RS7 Sportback', Brand:'Audi', count:523 }
];

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails,
 palette:['red','green'],
 weightValuePath: 'count',
 leafItemSettings: {
 labelPath: 'Car'
 }
}, '#container');

```

**INDEX.HTML**



```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

### Desaturation with multiple colors

Multiple colors are used as gradient effect to specific shapes based on the ranges in datasource. By using **color** property, you can set n number of colors.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 legendSettings:{
 visible:true
 },
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping:[
 {
 from:500,
 to:2500,

```

```

 color: ['orange', 'pink']
 },
 {
 from: 3000,
 to: 5000,
 color: ['green', 'red', 'blue']
 }
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Color for items excluded from color mapping

Get the excluded ranges from data source using the color mapping and apply the specific color to those items, without specifying the `from` and `to` properties.

## INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];

```

```
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 legendSettings: {
 visible: true
 },
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping: [
 {
 from: 500,
 to: 2500,
 color: 'orange'
 },
 {
 from: 3000,
 to: 4000,
 color: 'green'
 },
 {
 color: 'red'
 }
]
 }
}, '#container');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
```

```
</body></html>
```

### Bind the colors to the items from data source

To set the color for each item from the data source, bind the data source property to the **colorValuePath**.

#### INDEX.TS

```
let fruits:Object[] = [{ fruit:'Apple', count:5000, color: 'red'},
 { fruit:'Mango', count:3000, color:'blue'},
 { fruit:'Orange', count:2300, color:'green' },
 { fruit:'Banana', count:500, color:'yellow' },
 { fruit:'Grape', count:4300, color:'orange' },
 { fruit:'Papaya', count:1200, color:'pink' },
 { fruit:'Melon', count:4500, color:'violet' }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 colorValuePath: 'color',
 leafItemSettings: {
 labelPath: 'fruit',
 }
}, '#container');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## Data label in EJ2 JavaScript Treemap control

Data Labels are used to identify the name of items or groups in the TreeMap component. Data Labels will be shown by specifying the data source properties in the `labelPath` of the `leafItemSettings`.

### Format

Customize the labels for each item using the `labelFormat` property in the `leafItemSettings`.

### INDEX.TS

```
let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
 { Car:'RS7 Sportback', Brand:'Audi', count:523 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails,
 weightValuePath: 'count',
 leafItemSettings: {
 labelPath: 'Car',
 labelFormat: '${Car}-${Brand}'
 }
}, '#container');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Template

The template supports customizing labels of each leaf node using the `labelTemplate` property. It uses Essential JS2 template engine to render elements and the position of templates can be customize using the `templatePosition` property.

### INDEX.TS

```
let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
 { Car:'RS7 Sportback', Brand:'Audi', count:523 }
];

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails,
 weightValuePath: 'count',
 leafItemSettings: {
 labelPath: 'Car',
 labelTemplate: '<div>{{:Car}}-{{:Brand}}</div>',
 templatePosition: 'Center'
 }
}, '#container');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### InterSectAction

When the label size in each item exceeds the actual size, use the `interSectAction` property in the `leafItemSettings` to customise the labels.

### INDEX.TS

```

let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet Purvis Eureka LIoyd-Hartnett',
Brand:'Audi',count:123 },
 { car:'RS7 Sportback', Brand:'Audi', count:523 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails ,
 weightValuePath: 'count',
 leafItemSettings: {
 labelPath: 'Car',
 labelFormat: '${Car}-${Brand}',
 interSectAction: 'WrapByWord'
 }
}, '#container');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Legend in EJ2 JavaScript Treemap control

Legend is used to provide valuable information for interpreting what the TreeMap displays. The legends can be represented in various colors, shapes or other identifiers based on the data.

### Position and alignment

Legend position is used to place legend in various positions. Based on the legend position, the legend item will be aligned. For example, if the position is top or bottom, the legend items are placed by rows. If the position is left or right, the legend items are placed by columns.

The following options are available to customize the legend position:

- Top
- Bottom
- Left
- Right
- Float

### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];

import { TreeMap, TreeMapLegend } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapLegend);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 legendSettings: {
 visible: true,
 position: 'Top'
 },
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping:[
 {
 from:500,
 to:3000,
 color:'orange'
 },
 {
 from:3000,
 to:5000,
 color:'green'
 }
]
 }
});

```



```

 }
]
}
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

Legend Alignment is used to align the legend items in specific location. The following options are available to customize the legend alignment:

- Near
- Center
- Far

## INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap, TreeMapLegend } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapLegend);

```

```
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 legendSettings: {
 visible: true,
 alignment: 'Far',
 },
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping: [
 {
 from: 500,
 to: 3000,
 color: 'orange'
 },
 {
 from: 3000,
 to: 5000,
 color: 'green'
 }
]
 }
}, '#container');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## Legend mode

The TreeMap control supports two different types of legend rendering modes such as **Default** and **Interactive**.

### Default mode

In default mode, the legends have symbols with legend labels that are used to identify the items in the TreeMap.

#### INDEX.TS

```
let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
 { Car:'RS7 Sportback', Brand:'Audi', count:523 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails ,
 weightValuePath: 'count',
 equalColorValuePath:'Brand',
 legendSettings: {
 visible: true,
 position:'Top',
 border:{color:'black',width:2}
 },
 leafItemSettings: {
 labelPath: 'Car',
 colorMapping:[
 {
 value:'Ford',
 color:'green'
 },
 {
 value:'Audi',
 color:'red'
 },
 {
 value:'Maruti',
 color:'orange'
 }
]
 },
}, '#container');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Interactive mode

The legends can be made interactive with an arrow mark that indicates exact range color in the legend when the mouse hovers on the TreeMap item. Enable this option by setting the `mode` property in the `legendSettings` to **Interactive**.

### INDEX.TS

```

let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
 { Car:'RS7 Sportback', Brand:'Audi', count:523 }
];

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails,
 weightValuePath: 'count',
 equalColorValuePath: 'Brand',
 legendSettings: {
 visible: true,
 mode: 'Interactive',
 position: 'Top',
 border: { color: 'black', width: 2 }
 },
 leafItemSettings: {
 labelPath: 'Car',
 colorMapping: [
 {
 value: 'Ford',
 color: 'green'
 },
 {
 value: 'Audi',

```

```

 color: 'red'
 },
 {
 value: 'Maruti',
 color: 'orange'
 }
]
},
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Legend size

Customize the legend size by modifying the `height` and `width` properties in the `legendSettings`. It accepts values in both percentage and pixel.

## INDEX.TS

```

let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
 { car:'RS7 Sportback', Brand:'Audi', count:523 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';

```

```
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails ,
 weightValuePath: 'count',
 equalColorValuePath: 'Brand',
 legendSettings: {
 visible: true,
 height: '50px',
 width: '200px',
 position: 'Top',
 border: {color: 'black', width: 2}
 },
 leafItemSettings: {
 labelPath: 'Car',
 colorMapping: [
 {
 value: 'Ford',
 color: 'green'
 },
 {
 value: 'Audi',
 color: 'red'
 },
 {
 value: 'Maruti',
 color: 'orange'
 }
]
 },
}, '#container');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Paging support

Treemap support legend paging, if the legend items cannot be placed within the provided **height** and **width** of the legend.

### INDEX.TS

```

let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
 { Car:'RS7 Sportback', Brand:'Audi', count:523 }
];

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails,
 weightValuePath: 'count',
 equalColorValuePath: 'Brand',
 legendSettings: {
 visible: true,
 height: '50px',
 width: '100px',
 border: {color: 'black', width: 2}
 },
 leafItemSettings: {
 labelPath: 'Car',
 colorMapping: [
 {
 value: 'Ford',
 color: 'green'
 },
 {
 value: 'Audi',
 color: 'red'
 },
 {
 value: 'Maruti',
 color: 'orange'
 }
]
 }
}, '#container');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Legend for items excluded from color mapping

Based on the mapping ranges in the data source, get the excluded ranges from the color mapping, and show the legend with the excluded range values that are bound to the specific legend.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 legendSettings:{
 visible:true
 },
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping:[
 {
 from:500,
 to:2500,
 minOpacity:0.2,
 maxOpacity:0.5,
 color:'orange'
 },

```



```

 {
 from: 3000,
 to: 4000,
 minOpacity: 0.5,
 maxOpacity: 0.8,
 color: 'green'
 },
 {
 color: 'red'
 }
]
}
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Hide desired legend items

To enable or disable the desired legend item for each color mapping, set the `showLegend` property to `true` in the `colorMapping`.

## INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },

```

```

 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 rangeColorValuePath: 'count',
 legendSettings:{
 visible:true
 },
 leafItemSettings: {
 labelPath: 'fruit',
 colorMapping:[
 {
 from:500,
 to:2500,
 color:'orange',
 showLegend: true
 },
 {
 from:3000,
 to:4000,
 color:'green',
 showLegend: false
 },
 {
 color:'red',
 showLegend: true
 }
]
 }
}, '#container');
```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
</script>
```

```

var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Hide legend items based data source value

To enable or disable the legend visibility for each item through the data source, bind the appropriate data source field name to `showLegendPath` property in the `legendSettings`.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000, visibility: true ,
color: '#71B081' },
 { fruit:'Orange', count:3000, visibility: false,
color: '#5A9A77' },
 { fruit:'Grape', count:2300, visibility: true, color:
'#498770' },
 { fruit:'Banana', count:500, visibility: false,
color: '#39776C' },
 { fruit:'Mango', count:4300, visibility: true, color:
'#266665' },
 { fruit:'Papaya', count:1200, visibility: false,
color: '#124F5E' }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 colorValuePath: 'color',
 palette: [],
 legendSettings:{
 visible:true,
 showLegendPath:'visibility'
 },
 leafItemSettings: {
 labelPath: 'fruit',
 }
}, '#container');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Bind legend item text from data source

To show the legend item text from the data source, bind the property name from data source to the `valuePath` property in the `legendSettings`.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000, visibility: true ,
color: '#71B081' },
 { fruit:'Orange', count:3000, visibility: false,
color: '#5A9A77' },
 { fruit:'Grape', count:2300, visibility: true, color:
'#498770' },
 { fruit:'Banana', count:500, visibility: false,
color: '#39776C' },
 { fruit:'Mango', count:4300, visibility: true, color:
'#266665' },
 { fruit:'Papaya', count:1200, visibility: false,
color: '#124F5E' }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 colorValuePath: 'color',
 palette: [],
 legendSettings:{
 visible:true,
 valuePath:'fruit'
 },
 leafItemSettings: {
 labelPath: 'fruit',
 }
}, '#container');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 for Treemap </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Hide duplicate legend items

To enable or disable the duplicate legend items, set the `removeDuplicateLegend` property to **true** in the `legendSettings`.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000, visibility: true ,
color: '#71B081' },
 { fruit:'Grape', count:3000, visibility: false,
color: '#5A9A77' },
 { fruit:'Apple', count:2300, visibility: true, color:
'#498770' },
 { fruit:'Banana', count:500, visibility: false,
color: '#39776C' },
 { fruit:'Grape', count:4300, visibility: true, color:
'#266665' },
 { fruit:'Papaya', count:1200, visibility: false,
color: '#124F5E' }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 colorValuePath: 'color',
 palette: [],
 legendSettings:{
 visible:true,
 valuePath:'fruit',
 removeDuplicateLegend: true
 }
});

```

```

 },
 leafItemSettings: {
 labelPath: 'fruit',
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Legend Responsiveness

Use a responsive legend that switches positions between the right and the bottom based on the available height and width. To enable the responsive legend, set the **position** property to **Auto** in the **legendSettings** and the legend position is changed based on the available height and width.

## INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Grape', count:3000 },
 { fruit:'Apple', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 }
];

import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',

```

```

 palette:['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'],
 width: '700px',
 height: '500px',
 legendSettings:{
 visible:true,
 position:'Auto'
 },
 leafItemSettings: {
 labelPath: 'fruit',
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Drilldown in EJ2 JavaScript Treemap control

The TreeMap supports drill-down to expose the hierarchy, achieved by clicking a node. If an item is clicked in the TreeMap, it will be moved to the next level or sub level hierarchy and returned back to the previous level by clicking the node.

### Perform drill-down action

The TreeMap items can be drilled by setting the `enableDrillDown` property to **true**.

## INDEX.TS

```
let jobData:Object[] = [
```

```

 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
 EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
 EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
 EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
 EmployeesCount: 100 }]];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
 "#009688"],
 dataSource: jobData,
 weightValuePath: 'EmployeesCount',
 enableDrillDown: true,
 levels: [
 { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobDescription', border: { color: 'black', width: 0.5
 } },
 { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
]
}, '#container');

```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">

```



```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### On-demand data loading

All the child items are rendered during the normal drill-down process, and visible at the initial rendering of the TreeMap. But on-demand data loading, it will not render child items at initial rendering, and child nodes will be rendered during the drill-down process by setting the `drillDownView` property to **true**.

### INDEX.TS

```

let jobData:Object[] = [
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },

```

```

 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }]];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
"#009688"],
 dataSource: jobData,
 weightValuePath: 'EmployeesCount',
 enableDrillDown: true,
 drillDownView: true,
 levels: [
 { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobDescription', border: { color: 'black', width: 0.5
} },
 { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}

```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Breadcrumb support

TreeMap items are drilled, up to any level of parent using breadcrumb navigation and the level from root parent to current level is displayed at the top of item layout. It can be enabled by using the `enableBreadcrumb` property to **true** and customize the breadcrumb connector using the `breadcrumbConnector` property. By default, `-(hyphen)` is the connector.

### INDEX.TS

```

let jobData:Object[] = [
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
 EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
 EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
 JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
 EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
 EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
 EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
 JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
 EmployeesCount: 100 }];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 palette: ["#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0",
 "#009688"],
 dataSource: jobData,
 weightValuePath: 'EmployeesCount',

```

```

enableDrillDown: true,
enableBreadcrumb: true,
breadcrumbConnector: ' -> ',
levels: [
 { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobDescription', border: { color: 'black', width: 0.5 } },
 { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
]
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Tooltip in EJ2 JavaScript Treemap control

Tooltip is used to display details about the items in the TreeMap. When space constraints prevent us from displaying the information using Data Labels, the tooltip comes in handy.

### Default tooltip

The tooltip is not visible by default, to make it visible, set the `visible` property in the `tooltipSettings` to `true` and injecting the `TreeMapTooltip` module using the `TreeMap.Inject(TreeMapTooltip)`.

## INDEX.JS

```

var treemap = new ej.treemap.TreeMap({
 dataSource: [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },

```

```

 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
],
 weightValuePath: 'count',
 tooltipSettings: {
 visible: true
 },
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Format tooltip

The tooltip content is displayed by default based on the `weightValuePath`. In addition, to show more information in the tooltip, use the `format` property and define field from the data source as `${datafield}`.

## INDEX.JS

```

var treemap = new ej.treemap.TreeMap({
 dataSource: [
 { fruit: 'Apple', count: 5000 },

```

```

 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
],
 weightValuePath: 'count',
 tooltipSettings: {
 visible: true,
 format: 'Name: ${fruit} - TotalCount: ${count}'
 },
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Tooltip template

Tooltip can be rendered as a custom component using the `template` property in the `tooltipSettings` which accepts one or more UI elements as an input, that can be rendered as a part of the tooltip rendering. You can use `${datafield}` as placeholder in HTML element to display the values from data source.

## INDEX.JS

```

var treemap = new ej.treemap.TreeMap({
 dataSource: [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
],
 weightValuePath: 'count',
 tooltipSettings: {
 visible: true,
 template: '<div><p>Name: ${fruit}</p><p>Total Count: ${count}</p></div>'
 },
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Selection and highlight in EJ2 JavaScript Treemap control

### Selection

Selection is used to select a particular group or item to differentiate from other items. Each item or each group can be selected and deselected while interacting with the items. The corresponding Treemap items are also selected while tapping a specific legend item, and vice versa.

The `fill` property is used to change the selected item color. The `color` and the `width` properties are used to customize the selected item border, and the selection is enabled by using the `enable` property to `true` in the `selectionSettings`.

### INDEX.JS

```
var treemap = new ej.treemap.TreeMap({
 dataSource: [
 { dataType: "Import", type: "Animal products", product:
 "2010", sales: 20839332874 },
 { dataType: "Import", type: "Animal products", product:
 "2011", sales: 23098635589 },
 { dataType: "Import", type: "Chemical products", product:
 "2010", sales: 141637951510 },
 { dataType: "Import", type: "Chemical products", product:
 "2011", sales: 161550338209 },
 { dataType: "Import", type: "Base metals", product:
 "2010", sales: 86079439944 },
 { dataType: "Import", type: "Base metals", product:
 "2011", sales: 103821671535 },
 { dataType: "Import", type: "Textile articles", product:
 "2010", sales: 97126140830 },
 { dataType: "Import", type: "Textile articles", product:
 "2011", sales: 104980750811 },
 { dataType: "Export", type: "Animal products", product:
 "2010", sales: 15845503378 },
 { dataType: "Export", type: "Animal products", product:
 "2011", sales: 20650111620 },
 { dataType: "Export", type: "Chemical products", product:
 "2010", sales: 136100054087 },
 { dataType: "Export", type: "Chemical products", product:
 "2011", sales: 146341672411 },
 { dataType: "Export", type: "Base metals", product:
 "2010", sales: 59060592813 },
 { dataType: "Export", type: "Base metals", product:
 "2011", sales: 71785882641 },
 { dataType: "Export", type: "Textile articles", product:
 "2010", sales: 20982380561 },
 { dataType: "Export", type: "Textile articles", product:
 "2011", sales: 26016143783 }
],
 weightValuePath: 'sales',
 levels: [
 { groupPath: 'dataType', fill: '#c5e2f7', headerStyle: { size:
 '16px' }, headerAlignment: 'Center', groupGap: 5 },
 { groupPath: 'product', fill: '#a4d1f2', headerAlignment:
 'Center' , groupGap: 2 }
],
 leafItemSettings: {
 labelPath: 'type',
```



```

 fill: '#8ebfe2',
 labelPosition: 'Center'
 },
 selectionSettings: {
 enable: true,
 fill: '#58a0d3',
 border: { width: 0.3, color: 'black' },
 opacity: '1'
 },
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Highlight

Highlight is used to highlight an item or group from other items or groups. Each item or each group can be highlighted by hovering the mouse over the items. The corresponding Treemap items are also be highlighted while hovering over a specific legend item, and vice versa.

The `fill` property is used to change the highlighted item color. The `color` and the `width` properties are used to customize the highlighted item border, and the highlight is enabled by setting the `enable` property to `true` in the `highlightSettings`.

## INDEX.JS

```

var treemap = new ej.treemap.TreeMap({
 dataSource: [

```

```

 { dataType: "Import", type: "Animal products", product:
"2010", sales: 20839332874 },
 { dataType: "Import", type: "Animal products", product:
"2011", sales: 23098635589 },
 { dataType: "Import", type: "Chemical products", product:
"2010", sales: 141637951510 },
 { dataType: "Import", type: "Chemical products", product:
"2011", sales: 161550338209 },
 { dataType: "Import", type: "Base metals", product:
"2010", sales: 86079439944 },
 { dataType: "Import", type: "Base metals", product:
"2011", sales: 103821671535 },
 { dataType: "Import", type: "Textile articles", product:
"2010", sales: 97126140830 },
 { dataType: "Import", type: "Textile articles", product:
"2011", sales: 104980750811 },
 { dataType: "Export", type: "Animal products", product:
"2010", sales: 15845503378 },
 { dataType: "Export", type: "Animal products", product:
"2011", sales: 20650111620 },
 { dataType: "Export", type: "Chemical products", product:
"2010", sales: 136100054087 },
 { dataType: "Export", type: "Chemical products", product:
"2011", sales: 146341672411 },
 { dataType: "Export", type: "Base metals", product:
"2010", sales: 59060592813 },
 { dataType: "Export", type: "Base metals", product:
"2011", sales: 71785882641 },
 { dataType: "Export", type: "Textile articles", product:
"2010", sales: 20982380561 },
 { dataType: "Export", type: "Textile articles", product:
"2011", sales: 26016143783 }
],
 weightValuePath: 'sales',
 levels: [
 { groupPath: 'dataType', fill: '#c5e2f7', headerStyle: { size:
'16px' }, headerAlignment: 'Center', groupGap: 5 },
 { groupPath: 'product', fill: '#a4d1f2', headerAlignment:
'Center' , groupGap: 2 }
],
 leafItemSettings: {
 labelPath: 'type',
 fill: '#8ebfe2',
 labelPosition: 'Center'
 },
 highlightSettings: {
 enable: true,
 fill: '#71b0dd',
 border: { width: 0.3, color: 'black' },
 opacity: '1'
 }
}, '#container');

```

**INDEX.HTML**

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 for Treemap </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Print and export in EJ2 JavaScript Treemap control

### Print

To use the print functionality, we should set the [allowPrint](#) property to **true**. The rendered treemap can be printed directly from the browser by calling the method [print](#).

### INDEX.JS

```

var treemap = new ej.treemap.TreeMap({
 allowPrint: true,
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {
 labelPath: 'State',
 labelFormat: '$ {State}
${GDP} Trillion
(${percentage}
%) ',
 labelStyle: {
 color: '#000000'

```

```

 },
 border: {
 color: '#000000',
 width: 0.5
 },
 }
}, '#container');
document.getElementById('print').onclick = () => {
 treemap.print();
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <button id="print" type="button" width="15%" style="float:
right">Print</button>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Export

### Image Export

To use the image export functionality, we should set the [allowImageExport](#) property to **true**. The rendered treemap can be exported as an image using the [export](#) method. The method requires two parameters: image type and file name. The treemap can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

**INDEX.JS**

```

var treemap = new ej.treemap.TreeMap({
 allowImageExport: true,
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {
 labelPath: 'State',
 labelFormat: '${State}
$$${GDP} Trillion
(${percentage}
%) ',
 labelStyle: {
 color: '#000000'
 },
 border: {
 color: '#000000',
 width: 0.5
 },
 },
}, '#container');
document.getElementById('export').onclick = () => {
 treemap.export('PNG', 'TreeMap');
};

```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>

```

```

<button id="export" type="button" width="15%" style="float:
right">Export</button>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

We can get the image file as base64 string for the JPEG and PNG formats. The treemap can be exported to image as a base64 string using the [export](#) method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

### INDEX.JS

```

var treemap = new ej.treemap.TreeMap({
 allowImageExport: true,
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {
 labelPath: 'State',
 labelFormat: '$${State}
$$${GDP} Trillion
($${percentage}
%) ',
 labelStyle: {
 color: '#000000'
 },
 border: {
 color: '#000000',
 width: 0.5
 },
 },
}, '#container');
document.getElementById('export').onclick = () => {
 treemap.export('JPEG', 'TreeMap', null, false).then((data) => {
 var base64 = data;
 document.writeln(base64);
 });
};

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <button id="export" type="button" width="15%" style="float:
right">Export</button>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

### PDF Export

To use the PDF export functionality, we should set the [allowPdfExport](#) property to **true**. The rendered treemap can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and **0** indicates portrait and **1** indicates landscape.

### INDEX.JS

```

var treemap = new ej.treemap.TreeMap({
 allowPdfExport: true,
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 weightValuePath: 'GDP',
 leafItemSettings: {
 labelPath: 'State',

```

```

 labelFormat: '${State}
$$GDP Trillion
($percentage
%) ',
 labelStyle: {
 color: '#000000'
 },
 border: {
 color: '#000000',
 width: 0.5
 },
 }
}, '#container');
document.getElementById('export').onclick = () => {
 treemap.export("PDF", "TreeMap", 0);
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <button id="export" type="button" width="15%" style="float:
right">Export</button>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The exporting of the treemap as base64 string is not supported in the PDF export.

## Accessibility in EJ2 JavaScript TreeMap control

TreeMap has built-in accessibility features like screen reading and WAI-ARIA attributes.

### WAI-ARIA attributes

The TreeMap control follows the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the TreeMap control:



| Attributes | Purpose |

| --- | --- |

| **role=region** | It specifies the TreeMap areas that do not support interactive functions like selection and highlight. |

| **role=button** | It specifies the TreeMap areas where interactive functions such as selection and highlight are available. |

| **aria-label** | Provides an accessible name for the data labels, legend title, and legend item labels. |

### Screen reading in TreeMap

Accessibility in the TreeMap control ensures that all users, regardless of ability or disability, can use screen reading. The following TreeMap elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Data labels | Reads the labels displayed on leaf items of the TreeMap. |

| Legend title | Reads the title of the legend in the TreeMap. |

| Legend item label | Reads the label of the legend item in the TreeMap. |

### Ensuring accessibility

The TreeMap control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the TreeMap control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the TreeMap control with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript controls](#)

## Internationalization in EJ2 JavaScript Treemap control

The TreeMap control supports internationalization for the following elements:

- Data label
- Tooltip

For more information about number and date formatter, refer to [internationalization](#).

### Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales. Internationalization library is used to globalize number, date, and time values in the TreeMap control using the **format** property in the TreeMap.

### Numeric format

In the following code example, tooltip is globalized to Deutsch culture.

```
`ts
```

```
import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { loadCldr, Ajax, setCulture, setCurrencyCode } from '@syncfusion/ej2-base';
TreeMap.Inject(TreeMapTooltip);
loadCultureFiles("de");
setCulture("de");
setCurrencyCode('EUR');
let treemap: TreeMap = new TreeMap({
 locale: "de",
 format: "c",
 dataSource: [
 {State:"United States", GDP:17946, percentage:11.08, Rank:1},
 {State:"China", GDP:10866, percentage: 28.42, Rank:2},
 {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
 {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
 {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
 {State:"France", GDP:2421, percentage:-9.69, Rank:6},
 {State:"India", GDP:2073, percentage:13.65, Rank:7},
 {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
 {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
 {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
],
 tooltipSettings: {
 visible: true,
 },
 weightValuePath: 'GDP'
}, '#container');
function loadCultureFiles(name: string) {
 var files = ['currencies.json', 'numbers.json'];
 var loadCulture = function (prop: number) {
 let val: any, ajax: Ajax;
 // ajax = new Ajax('http://ej2.syncfusion.com/javascript/demos/' + 'src/common/cldr-data/main/' +
 name + '/' + files[prop], 'GET', false);
```

```

ajax = new Ajax('/node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
ajax.onSuccess = function (value: any) {
 val = value;
};
ajax.send();
loadCldr(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
}
}
,

```

### Right-to-left rendering

The TreeMap control supports right-to-left rendering for all its elements such as nodes, tooltip, data labels, and legends.

### Legend with Rtl support

If set the `enableRtl` property to **true**, then the legend icon will be rendered on the right and the legend text will be rendered on the left of the legend icon.

### INDEX.TS

```

let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232, color:
'#71B081' },
 { Car:'EcoSport', Brand:'Ford', count:121, color:
'#5A9A77' },
 { Car:'Swift', Brand:'Maruti', count:143, color:
'#498770' },
 { Car:'Baleno', Brand:'Maruti', count:454, color:
'#39776C' },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 ,
color: '#266665' },
 { Car:'A3 Cabriolet', Brand:'Audi',count:123, color:
'#124F5E' }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails ,
 weightValuePath: 'count',
 colorValuePath: 'color',
 enableRtl:true,
 legendSettings: {
 visible: true,
 position:'Top'
 },
 leafItemSettings: {
 labelPath: 'Car'
 },

```

```
}, '#container');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## Tooltip with Rtl support

If the `enableRtl` property is set to **true**, the tooltip data will be rendered in reverse direction.

## INDEX.TS

```
let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 palette:['#71B081', '#5A9A77', '#498770', '#39776C',
'#266665', '#124F5E'],
 tooltipSettings: {
 visible: true,
 format:'${count} : ${fruit}'
```

```

 },
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## Treemap Item Rendering Direction

The direction of TreeMap item is **TopLeftBottomRight** by default and customize the rendering direction of the TreeMap item by setting the **renderDirection** property.

The TreeMap can be rendered in the following directions:

- TopLeftBottomRight
- TopRightBottomLeft
- BottomRightTopLeft
- BottomLeftTopRight

The following example demonstrate, how to render the treemap in the RTL direction with **TopLeftBottomRight**.

## INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },

```

```

 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 palette:['#71B081','#5A9A77', '#498770', '#39776C',
 '#266665','#124F5E'],
 renderDirection:'TopLeftBottomRight',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following example demonstrate, how to render the treemap in the RTL direction with TopRightBottomLeft.

### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },

```

```

 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 palette:['#71B081','#5A9A77', '#498770', '#39776C',
 '#266665','#124F5E'],
 renderDirection:'TopRightBottomLeft',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');
```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

The following example demonstrate, how to render the treemap in the RTL direction with BottomRightTopLeft.

### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
```

```

 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 palette:['#71B081','#5A9A77', '#498770', '#39776C',
 '#266665','#124F5E'],
 renderDirection:'BottomRightTopLeft',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');
```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

The following example demonstrate, how to render the treemap in the RTL direction with BottomLeftTopRight.

### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
```



```

 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 palette: ['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'],
 renderDirection: 'BottomLeftTopRight',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Right-to-left rendering

The TreeMap control supports right-to-left rendering for all its elements such as nodes, tooltip, data labels, and legends.

### Legend with Rtl support

If you set the `enableRtl` property to true, then the legend icon will be rendered on the right and the legend text will be rendered on the left of the legend icon.

### INDEX.TS

```
let CarDetails:Object[] = [{ Car:'Mustang', Brand:'Ford', count:232, color:
'#71B081' },
 { Car:'EcoSport', Brand:'Ford', count:121, color:
'#5A9A77' },
 { Car:'Swift', Brand:'Maruti', count:143, color:
'#498770' },
 { Car:'Baleno', Brand:'Maruti', count:454, color:
'#39776C' },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545,
color: '#266665' },
 { Car:'A3 Cabriolet', Brand:'Audi',count:123, color:
'#124F5E' }
];
import { TreeMap } from '@syncfusion/ej2-treemap';
let treemap: TreeMap = new TreeMap({
 dataSource: CarDetails,
 weightValuePath: 'count',
 colorValuePath: 'color',
 enableRtl:true,
 legendSettings: {
 visible: true,
 position:'Top'
 },
 leafItemSettings: {
 labelPath: 'Car'
 },
}, '#container');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
```

```

var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Tooltip with Rtl support

If the `enableRtl` property is set to true, the tooltip data will be rendered in reverse direction.

The following example shows the format of the tooltip.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];

import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 palette:['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'],
 tooltipSettings: {
 visible: true,
 format:'${count} : ${fruit}'
 },
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>

```

```
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Treemap Item Rendering Direction

By default, the direction of tree map item is **TopLeftBottomRight**. You can customize the rendering direction of the tree map item by setting the **renderDirection** property.

The TreeMap can be rendered in the following four different directions.

**TopLeftBottomRight**

**TopRightBottomLeft**

**BottomRightTopLeft**

**BottomLeftTopRight**

The following example demonstrate how to render the treemap in the RTL direction with **TopLeftBottomRight**.

#### INDEX.TS

```
let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits,
 weightValuePath: 'count',
 palette:['#71B081', '#5A9A77', '#498770', '#39776C',
'#266665', '#124F5E'],
 renderDirection:'TopLeftBottomRight',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 for Treemap </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following example demonstrate how to render the treemap in the RTL direction with **TopRightBottomLeft**.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];

import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 palette:['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'],
 renderDirection:'TopRightBottomLeft',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 for Treemap </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following example demonstrate how to render the treemap in the RTL direction with **BottomRightTopLeft**.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];

import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 palette:['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'],
 renderDirection:'BottomRightTopLeft',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 for Treemap </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

The following example demonstrate how to render the treemap in the RTL direction with **BottomLeftTopRight**.

#### INDEX.TS

```

let fruits:Object[] = [{ fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];

import { TreeMap, TreeMapTooltip } from '@syncfusion/ej2-treemap';
TreeMap.Inject(TreeMapTooltip);
let treemap: TreeMap = new TreeMap({
 dataSource: fruits ,
 weightValuePath: 'count',
 palette:['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'],
 renderDirection:'BottomLeftTopRight',
 leafItemSettings: {
 labelPath: 'fruit'
 }
}, '#container');

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>Essential JS 2 for Treemap </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for treemap UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## Ej1 api migration in EJ2 JavaScript Treemap control

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

### Data Binding

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| DataSource | **Property:** *dataSource* <br/> <br/> var populationdata = [{ <br/> &#160; Continent: "Asia", <br/> &#160; Population: 1749046000 }], <br/> \$("#container").ejTreeMap({ <br/> dataSource: populationdata }) | **Property:** *dataSource* <br/> <br/> var populationdata = [{ Continent: "Asia", <br/> &#160; Population: 1749046000 }], <br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; dataSource: populationdata <br/> }); <br/> treemap.appendTo('#container'); |

### Appearance

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Layout | **Property:** *itemsLayoutMode* <br/> <br/> \$("#container").ejTreeMap({ <br/> itemsLayoutMode: "sliceanddiceauto" }) | **Property:** *layoutType* <br/> <br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; layoutType: 'SliceAndDiceAuto' <br/> }); <br/> treemap.appendTo('#container'); |

| Weight Value Path | **Property:** *weightValuePath* <br/> <br/> \$("#container").ejTreeMap({ <br/> weightValuePath: "Population" }) | **Property:** *weightValuePath* <br/> <br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; weightValuePath: 'Population' <br/> }); <br/> treemap.appendTo('#container'); |



|Range Color Value Path| **Property:** *colorValuePath*<br/><br/> \$("#container").ejTreeMap({ <br/> colorValuePath: "Continent" })|**Property:** *rangeColorValuePath*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; rangeColorValuePath: 'Continent' <br/> }); <br/> treemap.appendTo('#container');|

|Equal Color Value Path| Not Applicable|**Property:** *equalColorValuePath*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; equalColorValuePath: 'Asia' <br/> }); <br/> treemap.appendTo('#container');|

|Height| **Property:** *height*<br/><br/> \$("#container").ejTreeMap({ <br/> height: 50 })|**Property:** *height*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; height: '50px' <br/> }); <br/> treemap.appendTo('#container');|

|Width| **Property:** *width*<br/><br/> \$("#container").ejTreeMap({ <br/> width: 400 })|**Property:** *width*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; width: '400px' <br/> }); <br/> treemap.appendTo('#container');|

|Theme| Not Applicable|**Property:** *theme*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; theme: 'Highcontrast' <br/> }); <br/> treemap.appendTo('#container');|

|Localization| **Property:** *locale*<br/><br/> \$("#container").ejTreeMap({ <br/> locale: "en-US" })|**Property:** *locale*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; locale: 'en-US' <br/> }); <br/> treemap.appendTo('#container');|

|Palette Colors| **Property:** *paletteColorMapping.colors*<br/><br/> \$("#container").ejTreeMap({ <br/> paletteColorMapping: { colors: ['red','green'] } })|**Property:** *palette*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; palette: ['#C33764', '#AB3566'] <br/> }); <br/> treemap.appendTo('#container');|

|Margin| Not Applicable|**Property:** *margin*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; margin: { left: 5, top: 8 } <br/> }); <br/> treemap.appendTo('#container');|

|Resize| **Property:** *enableResize*<br/><br/> \$("#container").ejTreeMap({ <br/> enableResize: true })| Not Applicable|

|Responsive| **Property:** *isResponsive*<br/><br/> \$("#container").ejTreeMap({ <br/> isResponsive: true })| Not Applicable|

## Leaf Items

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Border Color| **Property:** *leafItemSettings.borderBrush*<br/><br/> \$("#container").ejTreeMap({ <br/> &#160; leafItemSettings: { showLabels: true, <br/> &#160; borderBrush: "blue" } <br/> })|**Property:** *leafItemSettings.border*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; leafItemSettings: { border: { color: 'white' } } <br/> }); <br/> treemap.appendTo('#container');|

|Border Width| **Property:** *leafItemSettings.borderThickness*<br/><br/> \$("#container").ejTreeMap({ <br/> &#160; leafItemSettings: { showLabels: true, <br/> &#160; borderThickness: 5 } <br/> })|**Property:** *leafItemSettings.border*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160; leafItemSettings: { border: { width: 5 } } <br/> }); <br/> treemap.appendTo('#container');|

|Gap Value| **Property:** *leafItemSettings.gap*<br/><br/> \$("#container").ejTreeMap({ <br/> &#160; leafItemSettings: { showLabels: true, <br/> &#160; gap: 5 } <br/> })|**Property:**

```

leafItemSettings.gap

 var treemap = new ej.treemap.TreeMap({

leafItemSettings: { gap: 5 }
 });
 treemap.appendTo('#container');|

| Leaf Item Label| Property: leafItemSettings.itemTemplate

 $("#container").ejTreeMap({

 leafItemSettings: { showLabels: true,
 itemTemplate: "template" }

})| Property: leafItemSettings.labelTemplate

 var treemap = new ej.treemap.TreeMap({

 leafItemSettings: { labelTemplate: 'template' }
 });
 treemap.appendTo('#container');|

| Leaf Label Path| Property: leafItemSettings.labelPath

 $("#container").ejTreeMap({

 leafItemSettings: { showLabels: true,
 labelPath: "GameName" }
 })| Property:
leafItemSettings.labelPath

 var treemap = new ej.treemap.TreeMap({

leafItemSettings: { labelPath: 'GameName' }
 });
 treemap.appendTo('#container');|

| Leaf Label Position| Property: leafItemSettings.labelPosition

 $("#container").ejTreeMap({

 leafItemSettings: { showLabels: true,
 labelPosition: "topcenter" }

})| Property: leafItemSettings.labelPosition

 var treemap = new ej.treemap.TreeMap({

 leafItemSettings: { labelPosition: 'Center' }
 });
 treemap.appendTo('#container');|

| Leaf Label Color| Not Applicable| Property: leafItemSettings.fill

 var treemap = new
ej.treemap.TreeMap({
 leafItemSettings: { fill: 'red' }
 });

treemap.appendTo('#container');|

| Random Colors| Not Applicable| Property: leafItemSettings.autoFill

 var treemap = new
ej.treemap.TreeMap({
 leafItemSettings: { autoFill: true }
 });

treemap.appendTo('#container');|

| Format| Not Applicable| Property: leafItemSettings.labelFormat

 var treemap = new
ej.treemap.TreeMap({
 leafItemSettings: { labelFormat: '${Continent}-${Population}' }

});
 treemap.appendTo('#container');|

| Labels Visibility| Property: leafItemSettings.showLabels

 $("#container").ejTreeMap({

 leafItemSettings: { showLabels: true }
 })| Property: leafItemSettings.showLabels

var treemap = new ej.treemap.TreeMap({
 leafItemSettings: { showLabels: false }
 });

 treemap.appendTo('#container');|

| Opacity| Not Applicable| Property: leafItemSettings.opacity

 var treemap = new
ej.treemap.TreeMap({
 leafItemSettings: { opacity: 0.7 }
 });

treemap.appendTo('#container');|

| Padding| Not Applicable| Property: leafItemSettings.padding

 var treemap = new
ej.treemap.TreeMap({
 leafItemSettings: { padding: 5 }
 });

treemap.appendTo('#container');|

| Font Customization| Not Applicable| Property: leafItemSettings.labelStyle

 var treemap =
new ej.treemap.TreeMap({
 leafItemSettings: { labelStyle: { size: '12px', color: 'red', opacity:
0.5 } }
 });
 treemap.appendTo('#container');|

| Position of Template| Not Applicable| Property: leafItemSettings.templatePosition

 var
treemap = new ej.treemap.TreeMap({
 leafItemSettings: { templatePosition: 'Center' }

 });
 treemap.appendTo('#container');|

```

## Legend

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Legend Alignment | **Property:** *legendSettings.alignment*  
 \$("#container").ejTreeMap({  
 legendSettings: { alignment: "far" }  
 }) | **Property:** *legendSettings.alignment* var treemap = new ej.treemap.TreeMap({  
 legendSettings: { alignment: 'Near' }  
 }); treemap.appendTo('#container');

| Legend Visibility | **Property:** *showLegend*  
 \$("#container").ejTreeMap({  
 showLegend: false  
 }) | **Property:** *legendSettings.visible* var treemap = new  
 ej.treemap.TreeMap({  
 legendSettings: { visible: true }  
 }); treemap.appendTo('#container');

| Legend Position | **Property:** *legendSettings.dockPosition*  
 \$("#container").ejTreeMap({  
 legendSettings: { dockPosition: "bottom" }  
 }) | **Property:** *legendSettings.position*  
 var treemap = new ej.treemap.TreeMap({  
 legendSettings: { position: 'Top' }  
 }); treemap.appendTo('#container');

| Legend Height | **Property:** *legendSettings.height*  
 \$("#container").ejTreeMap({  
 legendSettings: { height: 40 }  
 }) | **Property:** *legendSettings.height* var treemap = new  
 ej.treemap.TreeMap({  
 legendSettings: { height: '40px' }  
 }); treemap.appendTo('#container');

| Legend Width | **Property:** *legendSettings.width*  
 \$("#container").ejTreeMap({  
 legendSettings: { width: 100 }  
 }) | **Property:** *legendSettings.width* var treemap = new  
 ej.treemap.TreeMap({  
 legendSettings: { width: '100px' }  
 }); treemap.appendTo('#container');

| Shape Height | **Property:** *legendSettings.iconHeight*  
 \$("#container").ejTreeMap({  
 legendSettings: { iconHeight: 15 }  
 }) | **Property:** *legendSettings.shapeHeight* var  
 treemap = new ej.treemap.TreeMap({  
 legendSettings: { shapeHeight: '40px' }  
 });  
 treemap.appendTo('#container');

| Shape Width | **Property:** *legendSettings.iconWidth*  
 \$("#container").ejTreeMap({  
 legendSettings: { iconWidth: 8 }  
 }) | **Property:** *legendSettings.shapeWidth* var  
 treemap = new ej.treemap.TreeMap({  
 legendSettings: { shapeWidth: '40px' }  
 });  
 treemap.appendTo('#container');

| Padding | Not Applicable | **Property:** *legendSettings.shapePadding*  
 var treemap = new  
 ej.treemap.TreeMap({  
 legendSettings: { shapePadding: 10 }  
 }); treemap.appendTo('#container');

| Legend Title | **Property:** *legendSettings.title*  
 \$("#container").ejTreeMap({  
 legendSettings: { title: "Population" }  
 }) | **Property:** *legendSettings.title* var treemap =  
 new ej.treemap.TreeMap({  
 legendSettings: { title: 'Legend' }  
 }); treemap.appendTo('#container');

| Legend Shape | Not Applicable | **Property:** *legendSettings.shape*  
 var treemap = new  
 ej.treemap.TreeMap({  
 legendSettings: { shape: 'Rectangle' }  
 }); treemap.appendTo('#container');

| Legend Mode | **Property:** *legendSettings.mode*  
 \$("#container").ejTreeMap({  
 legendSettings: { mode: "interactive" }  
 }) | **Property:** *legendSettings.mode* var treemap =  
 new ej.treemap.TreeMap({  
 legendSettings: { mode: 'Interactive' }  
 }); treemap.appendTo('#container');

| Legend Text Customization | Not Applicable | **Property:** *legendSettings.textStyle*  
 = new ej.treemap.TreeMap({   
 &#160; legendSettings: { textStyle: {   
 &#160; &#160; size: '10px', opacity: 0.5, color: 'red' } }   
 });   
 treemap.appendTo('#container');

| Legend Title Customization | Not Applicable | **Property:** *legendSettings.titleStyle*  
 = new ej.treemap.TreeMap({   
 &#160; legendSettings: { titleStyle: {   
 &#160; &#160; size: '10px', opacity: 0.5, color: 'red' } }   
 });   
 treemap.appendTo('#container');

| Legend Shape Border | Not Applicable | **Property:** *legendSettings.shapeBorder*  
 = new ej.treemap.TreeMap({   
 &#160; legendSettings: { shapeBorder: {   
 &#160; &#160; width: 2, color: 'red' } }   
 });   
 treemap.appendTo('#container');

| Legend Template | **Property:** *legendSettings.template*  
 \$("#container").ejTreeMap({   
 &#160; legendSettings: { template: "template" }   
 }); | Not Applicable |

| Left Label | **Property:** *legendSettings.leftLabel*  
 \$("#container").ejTreeMap({   
 &#160; legendSettings: { mode: "interactive", leftLabel: "10Million" }   
 }); | Not Applicable |

| Right Label | **Property:** *legendSettings.rightLabel*  
 \$("#container").ejTreeMap({   
 &#160; legendSettings: { mode: "interactive", rightLabel: "100Million" }   
 }); | Not Applicable |

| Legend Shape Image | Not Applicable | **Property:** *legendSettings.imageUrl*  
 = new ej.treemap.TreeMap({   
 &#160; legendSettings: { imageUrl: "image.png" }   
 });   
 treemap.appendTo('#container');

| Position in Intractive Legend | Not Applicable | **Property:** *legendSettings.labelPosition*  
 = new ej.treemap.TreeMap({   
 &#160; legendSettings: { labelPosition: "Center" }   
 });   
 treemap.appendTo('#container');

| Legend Location | Not Applicable | **Property:** *legendSettings.location*  
 = new ej.treemap.TreeMap({   
 &#160; legendSettings: { location: { x: 10, y: 20 } }   
 });   
 treemap.appendTo('#container');

| Legend Orientation | Not Applicable | **Property:** *legendSettings.orientation*  
 = new ej.treemap.TreeMap({   
 &#160; legendSettings: { orientation: "Horizontal" }   
 });   
 treemap.appendTo('#container');

## Levels

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Random Colors | Not Applicable | **Property:** *levels.autoFill*  
 = new ej.treemap.TreeMap({   
 &#160; levels: [{ autoFill: true } ]   
 });   
 treemap.appendTo('#container');

| Level Background Color | **Property:** *levels.groupBackground*  
 \$("#container").ejTreeMap({   
 &#160; levels: [{ groupBackground: "white" } ]   
 }); | **Property:** *levels.fill*  
 = new ej.treemap.TreeMap({   
 &#160; levels: [{ fill: 'white' } ]   
 });   
 treemap.appendTo('#container');

| Level Border Color | **Property:** *levels.groupBorderColor*  
 \$("#container").ejTreeMap({   
 &#160; levels: [{ groupBorderColor: "#58585B" } ]   
 }); | **Property:** *levels.border.color*  
 = new ej.treemap.TreeMap({   
 &#160; levels: [{ border: { color: "#58585B" } } ]   
 });   
 treemap.appendTo('#container');

| Level Border Width| **Property:** *levels.groupBorderThickness*<br/><br/> *\$("#container").ejTreeMap({*  
 <br/> &#160; levels: [{ groupBorderThickness: 2 }] <br/> })| **Property:** *levels.border.width*<br/><br/> var  
 treemap = new ej.treemap.TreeMap({ <br/> &#160; levels: [{ border: { width: 2 }] } <br/> }); <br/>  
 treemap.appendTo('#container');|

| Group Gap| **Property:** *levels.groupGap*<br/><br/> *\$("#container").ejTreeMap({* <br/> &#160; levels: [{  
 groupGap: 2 }] <br/> })| **Property:** *levels.groupGap*<br/><br/> var treemap = new ej.treemap.TreeMap({  
 <br/> &#160; levels: [{ groupGap: 2 }] <br/> }); <br/> treemap.appendTo('#container');|

| Group Padding| **Property:** *levels.groupPadding*<br/><br/> *\$("#container").ejTreeMap({* <br/> &#160;  
 levels: [{ groupPadding: 1 }] <br/> })| **Property:** *levels.groupPadding*<br/><br/> var treemap = new  
 ej.treemap.TreeMap({ <br/> &#160; levels: [{ groupPadding: 1 }] <br/> }); <br/>  
 treemap.appendTo('#container');|

| Group Path| **Property:** *levels.groupPath*<br/><br/> *\$("#container").ejTreeMap({* <br/> &#160; levels: [{  
 groupPath: "pathname" }] <br/> })| **Property:** *levels.groupPath*<br/><br/> var treemap = new  
 ej.treemap.TreeMap({ <br/> &#160; levels: [{ groupPath: 'pathname' }] <br/> }); <br/>  
 treemap.appendTo('#container');|

| Height of Header Level| **Property:** *levels.headerHeight*<br/><br/> *\$("#container").ejTreeMap({* <br/>  
 &#160; levels: [{ headerHeight: 20 }] <br/> })| **Property:** *levels.headerHeight*<br/><br/> var treemap =  
 new ej.treemap.TreeMap({ <br/> &#160; levels: [{ headerHeight: 20 }] <br/> }); <br/>  
 treemap.appendTo('#container');|

| Header Template| **Property:** *levels.headerTemplate*<br/><br/> *\$("#container").ejTreeMap({* <br/>  
 &#160; levels: [{ headerTemplate: "template" }] <br/> })| **Property:** *levels.headerTemplate*<br/><br/>  
 var treemap = new ej.treemap.TreeMap({ <br/> &#160; levels: [{ headerTemplate: 'template' }] <br/> });  
 <br/> treemap.appendTo('#container');|

| Opacity of Color| Not Applicable| **Property:** *levels.opacity*<br/><br/> var treemap = new  
 ej.treemap.TreeMap({ <br/> &#160; levels: [{ opacity: 0.5 }] <br/> }); <br/>  
 treemap.appendTo('#container');|

| Header Visibility| **Property:** *levels.showHeader*<br/><br/> *\$("#container").ejTreeMap({* <br/> &#160;  
 levels: [{ showHeader: false }] <br/> })| **Property:** *levels.showHeader*<br/><br/> var treemap = new  
 ej.treemap.TreeMap({ <br/> &#160; levels: [{ showHeader: false }] <br/> }); <br/>  
 treemap.appendTo('#container');|

| Template Position| **Property:** *levels.labelPosition*<br/><br/> *\$("#container").ejTreeMap({* <br/> &#160;  
 levels: [{ labelPosition: "topleft" }] <br/> })| **Property:** *levels.templatePosition*<br/><br/> var treemap =  
 new ej.treemap.TreeMap({ <br/> &#160; levels: [{ templatePosition: 'Center' }] <br/> }); <br/>  
 treemap.appendTo('#container');|

| Header Style| Not Applicable| **Property:** *levels.headerStyle*<br/><br/> var treemap = new  
 ej.treemap.TreeMap({ <br/> &#160; levels: [{ headerStyle: { color: 'red', size: '16px', opacity: 0.7 } }]  
 <br/> }); <br/> treemap.appendTo('#container');|

| Header Format| Not Applicable| **Property:** *levels.headerFormat*<br/><br/> var treemap = new  
 ej.treemap.TreeMap({ <br/> &#160; levels: [{ headerFormat: '{Continent}' }] <br/> }); <br/>  
 treemap.appendTo('#container');|

| Header Alignment| Not Applicable| **Property:** *levels.headerAlignment*<br/><br/> var treemap = new  
 ej.treemap.TreeMap({ <br/> &#160; levels: [{ headerAlignment: 'Center' }] <br/> }); <br/>  
 treemap.appendTo('#container');|

## Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Selection | **Property:** *selectionMode*  
 \$("#container").ejTreeMap({   
 &#160; selectionMode: "default"   
 }) | **Property:** *selectionSettings.mode*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; selectionSettings: { enable: true, mode: 'Item' }   
 });   
 treemap.appendTo('#container');

| Selection Color | Not Applicable | **Property:** *selectionSettings.fill*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; selectionSettings: { enable: true, fill: 'blue' }   
 });   
 treemap.appendTo('#container');

| Selection Color Opacity | Not Applicable | **Property:** *selectionSettings.opacity*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; selectionSettings: { enable: true, fill: 'blue', opacity: 0.6 }   
 });   
 treemap.appendTo('#container');

| Border for selection | Not Applicable | **Property:** *selectionSettings.border*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; selectionSettings: { border: { color: 'red', width: 2 } }   
 });   
 treemap.appendTo('#container');

## Highlight

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Highlight Group Selection Mode | **Property:** *highlightGroupOnSelection*  
 \$("#container").ejTreeMap({   
 &#160; highlightGroupOnSelection: true   
 }) | **Property:** *highlightSettings.mode*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, mode: 'All' }   
 });   
 treemap.appendTo('#container');

| Highlight Selection Mode | **Property:** *highlightOnSelection*  
 \$("#container").ejTreeMap({   
 &#160; highlightOnSelection: true   
 }) | **Property:** *highlightSettings.mode*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, mode: 'Item' }   
 });   
 treemap.appendTo('#container');

| Highlight Group Border Color | **Property:** *highlightGroupBorderBrush*  
 \$("#container").ejTreeMap({   
 &#160; highlightGroupBorderBrush: 'gray'   
 }) | **Property:** *highlightSettings.border.color*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, mode: 'All',   
 &#160; &#160; border: { color: 'gray' } }   
 });   
 treemap.appendTo('#container');

| Highlight Group Border Width | **Property:** *highlightGroupBorderThickness*  
 \$("#container").ejTreeMap({   
 &#160; highlightGroupBorderThickness: 3   
 }) | **Property:** *highlightSettings.border.width*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, mode: 'All',   
 &#160; &#160; border: { width: 3 } }   
 });   
 treemap.appendTo('#container');

| Highlight Selection Border Color | **Property:** *highlightBorderBrush*  
 \$("#container").ejTreeMap({   
 &#160; highlightBorderBrush: 'gray'   
 }) | **Property:** *highlightSettings.border.color*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, mode: 'Item',   
 &#160; &#160; border: { color: 'gray' } }   
 });   
 treemap.appendTo('#container');



| Highlight Selection Border Width | **Property:** *highlightBorderThickness*  
 \$("#container").ejTreeMap({   
 &#160; highlightBorderThickness: 3   
 }) | **Property:** *highlightSettings.border.width*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, mode: 'Item',   
 &#160; &#160; border: { width: 3 } }   
 });   
 treemap.appendTo('#container');

| Highlight Color | Not Applicable | **Property:** *highlightSettings.fill*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, fill: 'red' }   
 });   
 treemap.appendTo('#container');

| Highlight Color Opacity | Not Applicable | **Property:** *highlightSettings.opacity*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; highlightSettings: { enable: true, fill: 'red', opacity: 0.5 }   
 });   
 treemap.appendTo('#container');

### Range ColorMapping

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| From value | **Property:** *rangeColorMapping.from*  
 \$("#container").ejTreeMap({   
 &#160; rangeColorMapping: [{ from: 1000 } ]   
 }) | **Property:** *leafItemSettings.colorMapping.from*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; leafItemSettings: { colorMapping: [{ from: 1000 } ] }   
 });   
 treemap.appendTo('#container');

| To value | **Property:** *rangeColorMapping.to*  
 \$("#container").ejTreeMap({   
 &#160; rangeColorMapping: [{ to: 100000 } ]   
 }) | **Property:** *leafItemSettings.colorMapping.to*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; leafItemSettings: { colorMapping: [{ to: 100000 } ] }   
 });   
 treemap.appendTo('#container');

| Color | **Property:** *rangeColorMapping.color*  
 \$("#container").ejTreeMap({   
 &#160; rangeColorMapping: [{ color: "#77D8D8" } ]   
 }) | **Property:** *leafItemSettings.colorMapping.color*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; leafItemSettings: { colorMapping: [{ color: "#77D8D8" } ] }   
 });   
 treemap.appendTo('#container');

| Legend Label | **Property:** *rangeColorMapping.legendLabel*  
 \$("#container").ejTreeMap({   
 &#160; rangeColorMapping: [{ legendLabel: "Growth" } ]   
 }) | **Property:** *leafItemSettings.colorMapping.label*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; leafItemSettings: { colorMapping: [{ label: "Growth" } ] }   
 });   
 treemap.appendTo('#container');

### Desaturation ColorMapping

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| From value | **Property:** *desaturationColorMapping.from*  
 \$("#container").ejTreeMap({   
 &#160; desaturationColorMapping: [{ from: 1000 } ]   
 }) | **Property:** *leafItemSettings.colorMapping.from*  
 var treemap = new ej.treemap.TreeMap({   
 &#160; leafItemSettings: { colorMapping: [{ from: 1000 } ] }   
 });   
 treemap.appendTo('#container');

| To value | **Property:** *desaturationColorMapping.to*  
 \$("#container").ejTreeMap({   
 &#160; desaturationColorMapping: [{ to: 10000 } ]   
 }) | **Property:**

*leafItemSettings.colorMapping.to*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160;  
leafItemSettings: { colorMapping: [{ to: 10000 }] } <br/> }); <br/> treemap.appendTo('#container');|

|Color| **Property:** *desaturationColorMapping.color*<br/><br/> \$("#container").ejTreeMap({ <br/>  
&#160; desaturationColorMapping: [{ color: "#77D8D8" }] <br/> })| **Property:**  
*leafItemSettings.colorMapping.color*<br/><br/> var treemap = new ej.treemap.TreeMap({ <br/> &#160;  
leafItemSettings: { colorMapping: [{ color: "#77D8D8" }] } <br/> }); <br/>  
treemap.appendTo('#container');|

|Value| Not Applicable| **Property:** *leafItemSettings.colorMapping.value*<br/><br/> var treemap = new  
ej.treemap.TreeMap({ <br/> &#160; leafItemSettings: { colorMapping: [{ value: "Population" }] } <br/> });  
<br/> treemap.appendTo('#container');|

|Minimum Opacity| Not Applicable| **Property:** *leafItemSettings.colorMapping.minOpacity*<br/><br/>  
var treemap = new ej.treemap.TreeMap({ <br/> &#160; leafItemSettings: { colorMapping: [{ minOpacity:  
0.7 }] } <br/> }); <br/> treemap.appendTo('#container');|

|Maximum Opacity| Not Applicable| **Property:** *leafItemSettings.colorMapping.maxOpacity*<br/><br/>  
var treemap = new ej.treemap.TreeMap({ <br/> &#160; leafItemSettings: { colorMapping: [{  
maxOpacity: 1 }] } <br/> }); <br/> treemap.appendTo('#container');|

## Tooltip

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Tooltip| **Property:** *showTooltip*<br/><br/> \$("#container").ejTreeMap({ <br/> &#160; showTooltip:  
true <br/> })| **Property:** *tooltipSettings.visible*<br/><br/> var treemap = new ej.treemap.TreeMap({  
<br/> &#160; tooltipSettings: { visible: true } <br/> }); <br/> treemap.appendTo('#container');|

|Tooltip Template| **Property:** *tooltipTemplate*<br/><br/> \$("#container").ejTreeMap({ <br/> &#160;  
tooltipTemplate: 'template' <br/> })| **Property:** *tooltipSettings.template*<br/><br/> var treemap = new  
ej.treemap.TreeMap({ <br/> &#160; tooltipSettings: { template: 'template' } <br/> }); <br/>  
treemap.appendTo('#container');|

|Tooltip Border| Not Applicable| **Property:** *tooltipSettings.border*<br/><br/> var treemap = new  
ej.treemap.TreeMap({ <br/> &#160; tooltipSettings: { border: { color: 'red', width: 2 } } <br/> }); <br/>  
treemap.appendTo('#container');|

|Tooltip Color| Not Applicable| **Property:** *tooltipSettings.fill*<br/><br/> var treemap = new  
ej.treemap.TreeMap({ <br/> &#160; tooltipSettings: { fill: 'gray' } <br/> }); <br/>  
treemap.appendTo('#container');|

|Tooltip Format| Not Applicable| **Property:** *tooltipSettings.format*<br/><br/> var treemap = new  
ej.treemap.TreeMap({ <br/> &#160; tooltipSettings: { format: '\${Population}' } <br/> }); <br/>  
treemap.appendTo('#container');|

|Tooltip Marker Shape| Not Applicable| **Property:** *tooltipSettings.markerShapes*<br/><br/> var treemap  
= new ej.treemap.TreeMap({ <br/> &#160; tooltipSettings: { markerShapes: 'Circle' } <br/> }); <br/>  
treemap.appendTo('#container');|

|Tooltip Color Opacity| Not Applicable| **Property:** *tooltipSettings.opacity*<br/><br/> var treemap = new  
ej.treemap.TreeMap({ <br/> &#160; tooltipSettings: { opacity: 0.5 } <br/> }); <br/>  
treemap.appendTo('#container');|



| Tooltip Text Style | Not Applicable | **Property:** *tooltipSettings.textStyle*  
 ej.treemap.TreeMap({ tooltipSettings: { textStyle: { Color: 'red', opacity: 0.5, size: '12px' } } }); treemap.appendTo('#container');

### Drilldown

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Drilldown | **Property:** *enableDrillDown*  
 enableDrillDown: true | **Property:** *enableDrillDown*  
 ej.treemap.TreeMap({ enableDrillDown: true }); treemap.appendTo('#container');

| Drilldown Level | **Property:** *drillDownLevel*  
 drillDownLevel: 1 | **Property:** *InitialDrillSettings.groupIndex*  
 ej.treemap.TreeMap({ InitialDrillSettings: { groupIndex: 1 } }); treemap.appendTo('#container');

### Methods

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Treemap Refresh Method | **Method:** *refresh*  
 refresh | **Method:** *refresh*  
 ej.treemap.TreeMap({ refresh: true }); treemap.appendTo('#container');

| Method to Drilldown | **Method:** *drillDown*  
 drillDown | **Method:** *drillDown*  
 ej.treemap.TreeMap({ drillDown: true }); treemap.appendTo('#container');

| Append to Method | Not Applicable | **Method:** *appendTo*  
 appendTo | **Method:** *appendTo*  
 ej.treemap.TreeMap({ appendTo: true }); treemap.appendTo('#container');

| Add Event Listener Method | Not Applicable | **Method:** *addEventListener*  
 addEventListener | **Method:** *addEventListener*  
 ej.treemap.TreeMap({ addEventListener: true }); treemap.appendTo('#container');

| Treemap Destroy Method | Not Applicable | **Method:** *destroy*  
 destroy | **Method:** *destroy*  
 ej.treemap.TreeMap({ destroy: true }); treemap.appendTo('#container');

| Treemap Exporting Method | Not Applicable | **Method:** *export*  
 export | **Method:** *export*  
 ej.treemap.TreeMap({ export: true }); treemap.appendTo('#container');

| Get the Module Name | Not Applicable | **Method:** *getModuleName*  
 getModuleName | **Method:** *getModuleName*  
 ej.treemap.TreeMap({ getModuleName: true }); treemap.appendTo('#container');

| Printing the Treemap | Not Applicable | **Method:** *print*  
 print | **Method:** *print*  
 ej.treemap.TreeMap({ print: true }); treemap.appendTo('#container');

| Resizing the Treemap | Not Applicable | **Method:** *resizeOnTreeMap*  
 resizeOnTreeMap | **Method:** *resizeOnTreeMap*  
 ej.treemap.TreeMap({ resizeOnTreeMap: true }); treemap.appendTo('#container');

| Inject Method (Tooltip) | Not Applicable | **Method:** *resizeOnTreeMap*  
 TreeMap.Inject(TreeMapTooltip);  
 var treemap = new ej.treemap.TreeMap({ });  
 treemap.appendTo('#container');

| Remove Event Listener Method | Not Applicable | **Method:** *removeEventListener*  
 var treemap = new ej.treemap.TreeMap({  
 treemap.removeEventListener();  
 treemap.appendTo('#container');

## Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Treemap Load Event | Not Applicable | **Event:** *load*  
 var treemap = new ej.treemap.TreeMap({  
 load: function(e: ILoadEventArgs): void { }  
 treemap.appendTo('#container');

| Treemap Loaded Event | Not Applicable | **Event:** *loaded*  
 var treemap = new ej.treemap.TreeMap({  
 loaded: function(e: ILoadedEventArgs): void { }  
 treemap.appendTo('#container');

| Event Before Print | Not Applicable | **Event:** *beforePrint*  
 var treemap = new ej.treemap.TreeMap({  
 beforePrint: function(e: IPrintEventArgs): void { }  
 treemap.appendTo('#container');

| Click Event | **Event:** *click*  
 \$("#container").ejTreeMap({  
 click: function (args) { }  
 treemap = new ej.treemap.TreeMap({  
 click: function(e: IItemClickEventArgs): void { }  
 treemap.appendTo('#container');

| Drill Start Event | **Event:** *drillStarted*  
 \$("#container").ejTreeMap({  
 drillStarted: function (args) { }  
 treemap = new ej.treemap.TreeMap({  
 drillStart: function(e: IDrillStartEventArgs): void { }  
 treemap.appendTo('#container');

| Drill End Event | Not Applicable | **Event:** *drillEnd*  
 var treemap = new ej.treemap.TreeMap({  
 drillEnd: function(e: IDrillEndEventArgs): void { }  
 treemap.appendTo('#container');

| Event on Item Click | Not Applicable | **Event:** *itemClick*  
 var treemap = new ej.treemap.TreeMap({  
 itemClick: function(e: IItemClickEventArgs): void { }  
 treemap.appendTo('#container');

| Treemap Item Select Event | **Event:** *treeMapItemSelected*  
 \$("#container").ejTreeMap({  
 treeMapItemSelected: function (args) { }  
 treemap = new ej.treemap.TreeMap({  
 itemSelected: function(e: IItemSelectedEventArgs): void { }  
 treemap.appendTo('#container');

| Treemap Item Rendering Event | **Event:** *itemRendering*  
 \$("#container").ejTreeMap({  
 itemRendering: function (args) { }  
 treemap = new ej.treemap.TreeMap({  
 itemRendering: function(e: IItemRenderingEventArgs): void { }  
 treemap.appendTo('#container');

| Treemap Item Move Event | Not Applicable | **Event:** *itemMove*  
 var treemap = new ej.treemap.TreeMap({  
 itemMove: function(e: IItemMoveEventArgs): void { }  
 treemap.appendTo('#container');

|Treemap Item Highlight Event| Not Applicable| **Event:** *itemHighlight*<br><br> var treemap = new ej.treemap.TreeMap({ <br> &#160; itemHighlight: function(e: IItemHighlightEventArgs): void { } <br> }); <br> treemap.appendTo('#container');|

|Template Header Render Event| **Event:** *headerTemplateRendering*<br><br> \$("#container").ejTreeMap({ <br> &#160; headerTemplateRendering: function (args) { } <br> })| Not Applicable|

|Drilldown Item Select Event| **Event:** *drillDownItemSelected*<br><br> \$("#container").ejTreeMap({ <br> &#160; drillDownItemSelected: function (args) { } <br> })| Not Applicable|

|Refresh Event| **Event:** *refreshed*<br><br> \$("#container").ejTreeMap({ <br> &#160; refreshed: function (args) { } <br> })| Not Applicable|

|Group Select Event| **Event:** *treeMapGroupSelected*<br><br> \$("#container").ejTreeMap({ <br> &#160; treeMapGroupSelected: function (args) { } <br> })| Not Applicable|

|Mouse Event| Not Applicable| **Event:** *mouseMove*<br><br> var treemap = new ej.treemap.TreeMap({ <br> &#160; mouseMove: function(e: IMouseMoveEventArgs): void { } <br> }); <br> treemap.appendTo('#container');|

|Resize Event| Not Applicable| **Event:** *resize*<br><br> var treemap = new ej.treemap.TreeMap({ <br> &#160; resize: function(e: IResizeEventArgs): void { } <br> }); <br> treemap.appendTo('#container');|

|Tooltip Render Event| Not Applicable| **Event:** *tooltipRendering*<br><br> var treemap = new ej.treemap.TreeMap({ <br> &#160; tooltipRendering: function(e: ITreeMapTooltipRenderEventArgs): void { } <br> }); <br> treemap.appendTo('#container');|

|Double Click Event| **Event:** *doubleClick*<br><br> \$("#container").ejTreeMap({ <br> &#160; doubleClick: function (args) { } <br> })| Not Applicable|

|Right Click Event| **Event:** *rightClick*<br><br> \$("#container").ejTreeMap({ <br> &#160; rightClick: function (args) { } <br> })| Not Applicable|

## How To

### Drilldown in EJ2 JavaScript Treemap control

#### *Customize the header for treemap drilldown*

You can add a header element as `<div>` and customize it to show the population of a particular country or continent on treemap drill-down.

To customize the header for treemap drill-down, follow the given steps:

#### **Step 1:**

Initialize the treemap and enable the drill-down option.

```
`javascript
// Initialize the treemap control
var treemap = new ej.treemap.TreeMap({
 palette: ['#9999ff', '#CCFF99', '#FFFF99', '#FF9999', '#FF99FF', '#FFCC66'],
 enableDrillDown: true,
 format: 'n',
```

```

useGroupingSeparator: true,
dataSource: DrillDown,
weightValuePath: 'Population',
tooltipSettings: {
 visible: true,
 format: '${Name} : ${Population}'
},
leafItemSettings: {
 labelPath: 'Name',
 showLabels: false,
 labelStyle: { size: '0px' },
 border: { color: 'black', width: 0.5 }
},
levels: [
 { groupPath: 'Continent', fill: '#336699', border: { color: 'black', width: 0.5 } },
 { groupPath: 'States', fill: '#336699', border: { color: 'black', width: 0.5 } },
 { groupPath: 'Region', showHeader: false, fill: '#336699', border: { color: 'black', width: 0.5 } },
],
});
// Render the initialized treemap
treemap.appendTo('#container');
`

```

**Step 2:**

Show the population of a particular continent in the treemap **loaded** event. In this event, you can get the header element.

```

`javascript
loaded: function (args) {
 var header = document.getElementById('header');
 var population = 0;
 for (var i = 0; i < args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
 population += +(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Population);
 }
 header.innerHTML = 'Continent - Population : ' + population
}

```

```

}
,

```

**Step 3:**

Customize the population for drilled countries or states in the header element when drill-down the treemap. The `drillEnd` event will be triggered when treemap is drilled.

**INDEX.JS**

```

var treemap = new ej.treemap.TreeMap(
{
 palette: ['#9999ff', '#CCFF99', '#FFFF99', '#FF9999', '#FF99FF',
 '#FFCC66'],
 enableDrillDown: true,
 format: 'n',
 useGroupingSeparator: true,
 dataSource: drillDown,
 weightValuePath: 'Population',
 tooltipSettings: {
 visible: true,
 format: '${Name} : ${Population}',
 },
 leafItemSettings: {
 labelPath: 'Name',
 showLabels: false,
 labelStyle: { size: '0px' },
 border: { color: 'black', width: 0.5 },
 },
 levels: [
 {
 groupPath: 'Continent',
 fill: '#336699',
 border: { color: 'black', width: 0.5 },
 },
 {
 groupPath: 'States',
 fill: '#336699',
 border: { color: 'black', width: 0.5 },
 },
 {
 groupPath: 'Region',
 showHeader: false,
 fill: '#336699',
 border: { color: 'black', width: 0.5 },
 },
],
 loaded: function (args) {
 var header = document.getElementById('header');
 var population = 0;
 for (var i = 0; i <
args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
 population +=
args.treemap.layout.renderItems[0]['parent'].Continent[i]['data']
 .Population;
 }
 }
}

```

```

 header.innerHTML = 'Continent - Population : ' + population;
 },
 drillEnd: function (args) {
 var header = document.getElementById('header');
 var layout = document.getElementById(
 'container_TreeMap_Squarified_Layout'
);
 var population = 0;
 if (args.treemap.layout.renderItems[0]['isDrilled']) {
 for (var i = 0; i < args.treemap.layout.renderItems.length;
i++) {
 population +=
args.treemap.layout.renderItems[i]['data'].Population;
 }
 header.innerHTML =
 layout.children[0].children[1].innerHTML.split(' ')[1] +
 ' - ' +
 population;
 } else if
(args.treemap.layout.renderItems[0]['parent'].Continent) {
 for (
 var i = 0;
 i <
args.treemap.layout.renderItems[0]['parent'].Continent.length;
 i++
) {
 population +=
args.treemap.layout.renderItems[0]['parent'].Continent[i]['data']
 .Population;
 }
 header.innerHTML = 'Continent - Population : ' + population;
 } else {
 population =
args.treemap.layout.renderItems[0]['data'].Population;
 header.innerHTML =
 layout.children[0].children[1].innerHTML.split(' ')[1] +
 ' - Population : ' +
 population;
 }
 },
 '#container'
);

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <script src="datasource.js"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
 <div id="header"></div>
 <div id="container">
 </div>
</script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Drilldown with label in EJ2 JavaScript Treemap control

You can add a label template as `<div>` element to the tree map control when using the label template. To add a label template to the tree map control, you have to hide another labels by setting the `showLabels` property to **false** in `leafItemSettings` to show only the label template.

To add label template to tree map drilldown, follow the given steps:

#### Step 1:

Create a tree map control and enable the drill-down option.

```

`javascript
var treemap = new ej.treemap.TreeMap({
dataSource: CarSales,
enableDrillDown: true,
weightValuePath: 'Sales',
palette: ["white"],
levels: [
{
groupPath: 'Continent',
border: { width: 0.5, color: 'black' }
},
{
groupPath: 'Company',
border: { width: 0.5, color: 'black' }
},
]
}

```

```
}, '#container');
```

## Step 2:

Add the label template in the `leafItemSettings` options, and then set the `showLabels` property to `false` to hide another labels and show only label template.

## INDEX.JS

```
var treemap = new ej.treemap.TreeMap({
 dataSource: [
 { Continent: "China", Company: "Volkswagen", Sales: 3005994 },
 { Continent: "China", Company: "General Motors", Sales: 1230044 },
 { Continent: "China", Company: "Honda", Sales: 1197023 },
 { Continent: "United States", Company: "General Motors", Sales:
3042775 },
 { Continent: "United States", Company: "Ford", Sales: 2599193 },
 { Continent: "United States", Company: "Toyota", Sales: 2449587 },
 { Continent: "Japan", Company: "Toyota", Sales: 1527977 },
 { Continent: "Japan", Company: "Honda", Sales: 706982 },
 { Continent: "Japan", Company: "Suzuki", Sales: 623041 },
 { Continent: "Germany", Company: "Volkswagen", Sales: 655977 },
 { Continent: "Germany", Company: "Mercedes", Sales: 310845 },
 { Continent: "Germany", Company: "BMW", Sales: 261931 },
 { Continent: "United Kingdom", Company: "Ford ", Sales: 319442 },
 { Continent: "United Kingdom", Company: "Vauxhall", Sales: 251146 },
 { Continent: "United Kingdom", Company: "Volkswagen", Sales: 206994
 }
],
 enableDrillDown: true,
 weightValuePath: 'Sales',
 drillStart: function(args) {
 var labelElementGroup =
document.getElementById('container_Label_Template_Group');
 labelElementGroup.remove();
 },
 palette: ["white"],
 leafItemSettings: {
 showLabels: false,
 labelTemplate: '<div style="background-color: red">{:Company}</div>',
 templatePosition: 'Center'
 },
 levels: [
 {
 groupPath: 'Continent',
 border: { width: 0.5, color: 'black' }
 },
 {
 groupPath: 'Company',
 border: { width: 0.5, color: 'black' }
 }
]
}, '#container');
```

## INDEX.HTML



```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for Treemap </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for treemap UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## TreeView

### Data binding in EJ2 JavaScript Treeview control

The TreeView component provides the option to load data either from local data sources or from remote data services. This can be done through `dataSource` property that is a member of the `fields` property. The `dataSource` property supports array of JavaScript objects and `DataManager`. It also supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of `DataManager` adaptors.

TreeView has `load on demand` (Lazy load), by default. It reduces the bandwidth size when consuming huge data. It loads first level nodes initially, and when parent node is expanded, loads the child nodes based on the `parentID/child` member.

By default, the `loadOnDemand` is set to true. By disabling this property, all the tree nodes are rendered at the beginning itself.

You can use the `dataBound` event to perform actions. This event will be triggered once the data source is populated in the TreeView.

### Local data

To bind local data to the TreeView, you can assign a JavaScript object array to the `dataSource` property. The TreeView component requires three fields (ID, text, and parentID) to render local data source. When mapper fields are not specified, it takes the default values as the mapping fields. Local data source can also be provided as an instance of the `DataManager`. It supports two kinds of local data binding methods.

- Hierarchical data
- Self-referential data

### Hierarchical data

TreeView can be populated with hierarchical data source that contains nested array of JSON objects. You can directly assign hierarchical data to the [dataSource](#) property, and map all the field members with corresponding keys from the hierarchical data to **fields** property.

In the following example, **code**, **name**, and **countries** columns from hierarchical data have been mapped to **id**, **text**, and **child** fields, respectively.

### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
//define the nested array of JSON objects
let continents: { [key: string]: Object; }[] = [
 {
 code: 'AF', name: 'Africa', countries: [
 { code: 'NGA', name: 'Nigeria' },
 { code: 'EGY', name: 'Egypt' },
 { code: 'ZAF', name: 'South Africa' }
]
 },
 {
 code: 'AS', name: 'Asia', expanded: true, countries: [
 { code: 'CHN', name: 'China' },
 { code: 'IND', name: 'India', selected: true },
 { code: 'JPN', name: 'Japan' }
]
 },
 {
 code: 'EU', name: 'Europe', countries: [
 { code: 'DNK', name: 'Denmark' },
 { code: 'FIN', name: 'Finland' },
 { code: 'AUT', name: 'Austria' }
]
 },
 {
 code: 'NA', name: 'North America', countries: [
 { code: 'USA', name: 'United States of America' },
 { code: 'CUB', name: 'Cuba' },
 { code: 'MEX', name: 'Mexico' }
]
 },
 {
 code: 'SA', name: 'South America', countries: [
 { code: 'BRA', name: 'Brazil' },
 { code: 'COL', name: 'Colombia' },
 { code: 'ARG', name: 'Argentina' }
]
 },
 {
 code: 'OC', name: 'Oceania', countries: [
 { code: 'AUS', name: 'Australia' },

```

```

 { code: 'NZL', name: 'New Zealand' },
 { code: 'WSM', name: 'Samoa' }
]
},
{
 code: 'AN', name: 'Antarctica', countries: [
 { code: 'BVT', name: 'Bouvet Island' },
 { code: 'ATF', name: 'French Southern Lands' }
]
},
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: continents, id: 'code', text: 'name', child:
'countries' }
});
treeObj.appendTo('#tree');
```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Self-referential data

TreeView can be populated from self-referential data structure that contains array of JSON objects with **parentID** mapping.

You can directly assign self-referential data to the **dataSource** property, and map all the field members with corresponding keys from self-referential data to **fields** property.

To render the root level nodes, specify the **parentID** as null or no need to specify the **parentID** in **dataSource**.

In the following example, **id**, **pid**, **hasChild**, and **name** columns from self-referential data have been mapped to **id**, **parentID**, **hasChildren**, and **text** fields, respectively.

### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
let localData: { [key: string]: Object }[] = [
 { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'Hot Singles' },
 { id: 3, pid: 1, name: 'Rising Artists' },
 { id: 4, pid: 1, name: 'Live Music' },
 { id: 6, pid: 1, name: 'Best of 2017 So Far' },
 { id: 7, name: 'Sales and Events', hasChild: true },
 { id: 8, pid: 7, name: '100 Albums - $5 Each' },
 { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
 { id: 10, pid: 7, name: 'CD Deals' },
 { id: 11, name: 'Categories', hasChild: true },
 { id: 12, pid: 11, name: 'Songs' },
 { id: 13, pid: 11, name: 'Bestselling Albums' },
 { id: 14, pid: 11, name: 'New Releases' },
 { id: 15, pid: 11, name: 'Bestselling Songs' },
 { id: 16, name: 'MP3 Albums', hasChild: true },
 { id: 17, pid: 16, name: 'Rock' },
 { id: 18, pid: 16, name: 'Gospel' },
 { id: 19, pid: 16, name: 'Latin Music' },
 { id: 20, pid: 16, name: 'Jazz' },
 { id: 21, name: 'More in Music', hasChild: true },
 { id: 22, pid: 21, name: 'Music Trade-In' },
 { id: 23, pid: 21, name: 'Redeem a Gift Card' },
 { id: 24, pid: 21, name: 'Band T-Shirts' },
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: localData, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' }
});
treeObj.appendTo('#tree');
```

### INDEX.HTML

```
<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```

</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Remote data

TreeView can also be populated from a remote data service with the help of [DataManager](#) component and [Query](#) property.

It supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of [DataManager](#) adaptors.

You can assign service data as an instance of [DataManager](#) to the [dataSource](#) property. To interact with remote data source, you have to provide the endpoint url.

The [DataManager](#) that acts as an interface between the service endpoint and the TreeView requires the following information to interact with service endpoint properly.

- [DataManager->url](#): Defines the service endpoint to fetch data.
- [DataManager->adaptor](#): Defines the adaptor option. By default, [ODataAdaptor](#) is used for remote binding.

Adaptor is responsible for processing response and request from/to the service endpoint. The [@syncfusion/ej2-data](#) package provides some predefined adaptors designed to interact with service endpoints. They are,

- [UrlAdaptor](#): Used to interact with remote services. This is the base adaptor for all remote based adaptors.

- **ODataAdaptor**: Used to interact with OData endpoints.
- **ODataV4Adaptor**: Used to interact with OData V4 endpoints.
- **WebApiAdaptor**: Used to interact with Web API created under OData standards.
- **WebMethodAdaptor**: Used to interact with web methods.

In the following example, **ODataV4Adaptor** is used to fetch data from remote services. The **EmployeeID**, **FirstName**, and **EmployeeID**

columns from Employees table have been mapped to **id**, **text**, and **hasChildren** fields respectively for first level nodes.

The **OrderID**, **EmployeeID**, and **ShipName** columns from orders table have been mapped to **id**, **parentID**, and **text** fields respectively for second level nodes.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
 url: 'https://services.odata.org/V4/Northwind/Northwind.svc',
 adaptor: new ODataV4Adaptor,
 crossDomain: true,
});
let query: Query = new
Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
let query1: Query = new
Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
let treeObj: TreeView = new TreeView({
 fields: { dataSource: data, query: query, id: 'EmployeeID', text:
'FirstName', hasChildren: 'EmployeeID', tooltip: 'Title',
 child: { dataSource: data, query: query1, id: 'OrderID', parentID:
'EmployeeID', text: 'ShipName' }
 }
});
treeObj.appendTo('#tree');
```

#### INDEX.HTML

```
<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Check box in EJ2 JavaScript Treeview control

The TreeView component allows you to check more than one node in TreeView without affecting the UI's appearance by enabling the [showCheckBox](#) property. When this property is enabled, checkbox appears before each TreeView node text.

- If one of the child nodes is not in a checked state, then the parent node will be in an intermediate state.
- If all the child nodes are in checked state, then the parent node's state will also be checked.
- If a parent node is checked, then all the child nodes' state will also be checked.

By default, the checkbox state of parent and child nodes are dependent on each other. If you need independent checked state, you can achieve it using the [autoCheck](#) property.

Using the [checkedNodes](#) property, you can set the nodes that need to be checked or get the ID of nodes that are currently checked in the TreeView component.

If you need to prevent the node check action for a particular node, the [nodeChecking](#) event can be used which is triggered before the TreeView node is checked/unchecked. The [nodeChecked](#) event will be triggered when the TreeView node is checked/unchecked successfully.

In the following example, the `showCheckBox` property is enabled.

### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'Australia', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'New South Wales', isChecked: true },
 { id: 3, pid: 1, name: 'Victoria' },
 { id: 4, pid: 1, name: 'South Australia' },
 { id: 6, pid: 1, name: 'Western Australia', isChecked: true },

```

```

 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'China', hasChild: true },
 { id: 12, pid: 11, name: 'Guangzhou' },
 { id: 13, pid: 11, name: 'Shanghai' },
 { id: 14, pid: 11, name: 'Beijing' },
 { id: 15, pid: 11, name: 'Shantou' },
 { id: 16, name: 'France', hasChild: true },
 { id: 17, pid: 16, name: 'Pays de la Loire' },
 { id: 18, pid: 16, name: 'Aquitaine' },
 { id: 19, pid: 16, name: 'Brittany' },
 { id: 20, pid: 16, name: 'Lorraine' },
 { id: 21, name: 'India', hasChild: true },
 { id: 22, pid: 21, name: 'Assam' },
 { id: 23, pid: 21, name: 'Bihar' },
 { id: 24, pid: 21, name: 'Tamil Nadu' },
 { id: 25, pid: 21, name: 'Punjab' }
];
 let treeObj: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', parentID: 'pid', text:
 'name', hasChildren: 'hasChild' },
 showCheckBox: true,
 });
 treeObj.appendTo('#tree');

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>

```



```

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Checked nodes

You can get or set the checked nodes in TreeView at initial rendering and dynamically by using the [checkedNodes](#) property. It returns the checked nodes' ID as an array.

In the following example, the **New South Wales** and **Western Australia** nodes are checked at initial rendering. If any more nodes are checked, the checked nodes' IDs will be displayed in alert.

### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView, NodeCheckEventArgs } from '@syncfusion/ej2-navigations';
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'Australia', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'New South Wales' },
 { id: 3, pid: 1, name: 'Victoria' },
 { id: 4, pid: 1, name: 'South Australia' },
 { id: 6, pid: 1, name: 'Western Australia' },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'China', hasChild: true },
 { id: 12, pid: 11, name: 'Guangzhou' },
 { id: 13, pid: 11, name: 'Shanghai' },
 { id: 14, pid: 11, name: 'Beijing' },
 { id: 15, pid: 11, name: 'Shantou' },
 { id: 16, name: 'France', hasChild: true },
 { id: 17, pid: 16, name: 'Pays de la Loire' },
 { id: 18, pid: 16, name: 'Aquitaine' },
 { id: 19, pid: 16, name: 'Brittany' },
 { id: 20, pid: 16, name: 'Lorraine' },
 { id: 21, name: 'India', hasChild: true },
 { id: 22, pid: 21, name: 'Assam' },
 { id: 23, pid: 21, name: 'Bihar' },
 { id: 24, pid: 21, name: 'Tamil Nadu' },
 { id: 25, pid: 21, name: 'Punjab' }
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', parentID: 'pid', text:
 'name', hasChildren: 'hasChild' },
 showCheckBox: true,
 checkedNodes: ['2', '6'],
 nodeChecked: nodeChecked
});
treeObj.appendTo('#tree');
function nodeChecked(args: NodeCheckEventArgs) {

```

```

 alert("The checked node's id: " + treeObj.checkedNodes); // To alert the
 checked node's id.
}

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## See Also

- [How to check/uncheck the checkbox on clicking the tree node text](#)

## Node editing in EJ2 JavaScript Treeview control

The TreeView allows you to edit nodes by setting the [allowEditing](#) property to **true**.

To directly edit the nodes in place, **double click** the TreeView node or **select** the node and press **F2** key.

When editing is completed by focus out or by pressing the **Enter** key, the modified node's text saves automatically. If you do not want to save the modified node's text in TreeView node, press **Escape** key. It does not save the edited text to the TreeView node.

- Node editing can also be performed programmatically by using the [beginEdit](#) method. On passing the node ID or element through this method, the edit textbox will be created for the particular node thus allowing us to edit it.
- If you need to validate or prevent editing, the [nodeEditing](#) event can be used which is triggered before the TreeView node is renamed. On successfully renaming a node the [nodeEdited](#) event will be triggered.

In the following example, the first level node's text cannot be changed, but all other level nodes' text can be changed.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView, NodeEditEventArgs } from '@syncfusion/ej2-navigations';
let hierarchicalData: { [key: string]: Object }[] = [
 { id: '01', name: 'Local Disk (C:)', expanded: true,
 subChild: [
 {
 id: '01-01', name: 'Program Files',
 subChild: [
 { id: '01-01-01', name: '7-Zip' },
 { id: '01-01-02', name: 'Git' },
 { id: '01-01-03', name: 'IIS Express' },
]
 },
 {
 id: '01-02', name: 'Users', expanded: true,
 subChild: [
 { id: '01-02-01', name: 'Smith' },
 { id: '01-02-02', name: 'Public' },
 { id: '01-02-03', name: 'Admin' },
]
 },
 {
 id: '01-03', name: 'Windows',
 subChild: [
 { id: '01-03-01', name: 'Boot' },
 { id: '01-03-02', name: 'FileManager' },
 { id: '01-03-03', name: 'System32' },
]
 }
]
 },
 {
 id: '02', name: 'Local Disk (D:)',
 subChild: [
 {
 id: '02-01', name: 'Personals',
 subChild: [
 { id: '02-01-01', name: 'My photo.png' },
 { id: '02-01-02', name: 'Rental document.docx' },
 { id: '02-01-03', name: 'Pay slip.pdf' },
]
 }
]
 }
]
```

```

 id: '02-02', name: 'Projects',
 subChild: [
 { id: '02-02-01', name: 'ASP Application' },
 { id: '02-02-02', name: 'TypeScript Application' },
 { id: '02-02-03', name: 'React Application' },
]
 },
 {
 id: '02-03', name: 'Office',
 subChild: [
 { id: '02-03-01', name: 'Work details.docx' },
 { id: '02-03-02', name: 'Weekly report.docx' },
 { id: '02-03-03', name: 'Wish list.csv' },
]
 },
]
},
{
 id: '03', name: 'Local Disk (E:)', icon: 'folder',
 subChild: [
 {
 id: '03-01', name: 'Pictures',
 subChild: [
 { id: '03-01-01', name: 'Wind.jpg' },
 { id: '03-01-02', name: 'Stone.jpg' },
 { id: '03-01-03', name: 'Home.jpg' },
]
 },
 {
 id: '03-02', name: 'Documents',
 subChild: [
 { id: '03-02-01', name: 'Environment Pollution.docx' },
 { id: '03-02-02', name: 'Global Warming.ppt' },
 { id: '03-02-03', name: 'Social Network.pdf' },
]
 },
 {
 id: '03-03', name: 'Study Materials',
 subChild: [
 { id: '03-03-01', name: 'UI-Guide.pdf' },
 { id: '03-03-02', name: 'Tutorials.zip' },
 { id: '03-03-03', name: 'TypeScript.7z' },
]
 },
],
}
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: hierarchicalData, id: 'id', text: 'name', child:
'subChild' },
 allowEditing: true,
 nodeEditing: editing
});
treeObj.appendTo('#tree');
function editing(args: NodeEditEventArgs) {
 //check whether node is root node or not
 if (args.node.parentNode.parentNode.nodeName !== "LI") {

```

```
 args.cancel = true;
 }
}
```

## INDEX.HTML

```
<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
</body></html>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
```

## See Also

- [How to validate the text when renaming the tree node](#)
- [How to process the tree node operations using context menu](#)

## Multiple selection in EJ2 JavaScript Treeview control

Selection provides an interactive support and highlights the node that you select. Selection can be done through simple mouse down or keyboard interaction.

The TreeView also supports selection of multiple nodes by setting [allowMultiSelection](#) to **true**.

To multi-select, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following example, the `allowMultiSelection` property is enabled.

Multi selection is not applicable through touch interactions.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'Australia', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'New South Wales', isSelected: true },
 { id: 3, pid: 1, name: 'Victoria' },
 { id: 4, pid: 1, name: 'South Australia' },
 { id: 6, pid: 1, name: 'Western Australia', isSelected: true },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'China', hasChild: true },
 { id: 12, pid: 11, name: 'Guangzhou' },
 { id: 13, pid: 11, name: 'Shanghai' },
 { id: 14, pid: 11, name: 'Beijing' },
 { id: 15, pid: 11, name: 'Shantou' },
 { id: 16, name: 'France', hasChild: true },
 { id: 17, pid: 16, name: 'Pays de la Loire' },
 { id: 18, pid: 16, name: 'Aquitaine' },
 { id: 19, pid: 16, name: 'Brittany' },
 { id: 20, pid: 16, name: 'Lorraine' },
 { id: 21, name: 'India', hasChild: true },
 { id: 22, pid: 21, name: 'Assam' },
 { id: 23, pid: 21, name: 'Bihar' },
 { id: 24, pid: 21, name: 'Tamil Nadu' },
 { id: 25, pid: 21, name: 'Punjab' }
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', parentID: 'pid', text:
 'name', hasChildren: 'hasChild', selected: 'isSelected' },
 allowMultiSelection: true,
});
treeObj.appendTo('#tree');
```

#### INDEX.HTML

```
<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<meta name="description" content="Essential JS 2 for TreeView UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Selected nodes

You can get or set the selected nodes in TreeView at initial rendering and dynamically by using the [selectedNodes](#) property. It will return the selected node's ID as an array.

- The [nodeselecting](#) event is triggered before a node is selected/unselected which can be used to prevent the selection.
- The [nodeSelected](#) event is triggered once a node is successfully selected/unselected.

In the following example, **New South Wales** and **Western Australia** nodes are selected at initial rendering. When a node is selected, the selected node's ID is displayed in alert.

### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView, NodeSelectEventArgs } from '@syncfusion/ej2-navigations';
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'Australia', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'New South Wales', isSelected: true },
 { id: 3, pid: 1, name: 'Victoria' },
 { id: 4, pid: 1, name: 'South Australia' },
 { id: 6, pid: 1, name: 'Western Australia', isSelected: true },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'China', hasChild: true },
 { id: 12, pid: 11, name: 'Guangzhou' },
 { id: 13, pid: 11, name: 'Shanghai' },
 { id: 14, pid: 11, name: 'Beijing' },

```

```

 { id: 15, pid: 11, name: 'Shantou' },
 { id: 16, name: 'France', hasChild: true },
 { id: 17, pid: 16, name: 'Pays de la Loire' },
 { id: 18, pid: 16, name: 'Aquitaine' },
 { id: 19, pid: 16, name: 'Brittany' },
 { id: 20, pid: 16, name: 'Lorraine' },
 { id: 21, name: 'India', hasChild: true },
 { id: 22, pid: 21, name: 'Assam' },
 { id: 23, pid: 21, name: 'Bihar' },
 { id: 24, pid: 21, name: 'Tamil Nadu' },
 { id: 25, pid: 21, name: 'Punjab' }
];
 let treeObj: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', parentID: 'pid', text:
 'name', hasChildren: 'hasChild' },
 allowMultiSelection: true,
 selectedNodes: ['2', '6'],
 nodeSelected: nodeSelected
 });
 treeObj.appendTo('#tree');
 function nodeSelected(args: NodeSelectEventArgs) {
 alert("The selected node's id: " + treeObj.selectedNodes); // To alert
 the selected node's id.
 }

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
</body>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}

```



```

}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Drag and drop in EJ2 JavaScript Treeview control

The TreeView component allows you to drag and drop any node by setting [allowDragAndDrop](#) to **true**. Nodes can be dragged and dropped at all levels of the same TreeView.

The dragged nodes can be dropped at any level by indicator lines with **line**, **plus/minus**, and **restrict** icons. It represents the exact position where the node is to be dropped as sibling or child.

The following table explains the usage of indicator icons.

Icons	Description
----- -----	
Plus icon	Indicates that the dragged node is to be added as child of target node.
Minus or restrict icon	Indicates that the dragged node is not to be dropped at the hovered region.
In between icon	Indicates that the dragged node is to be added as siblings of hovered region.

- If you need to prevent dragging action for a particular node, the [nodeDragStart](#) event can be used which is triggered when the node drag is started. If you need to prevent dropping action for a particular node, the [nodeDragStop](#) event can be used which is triggered when the drag is stopped.
- The [nodeDragging](#) event is triggered when the TreeView node is being dragged. You can customize the cloned element in this event.
- The [nodeDropped](#) event is triggered when the TreeView node is dropped on the target element successfully.

In the following sample, the [allowDragAndDrop](#) property is enabled.

#### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
let productTeam: { [key: string]: Object }[] = [
 {
 id: 1, name: 'ASP.NET MVC Team', expanded: true,
 child: [
 { id: 2, pid: 1, name: 'Smith' },
 { id: 3, pid: 1, name: 'Johnson', isSelected: true },
 { id: 4, pid: 1, name: 'Anderson' },
]
 },
 {
 id: 5, name: 'Windows Team',
 child: [
 { id: 6, pid: 5, name: 'Clark' },
 { id: 7, pid: 5, name: 'Wright' },
 { id: 8, pid: 5, name: 'Lopez' },
]
 }
]

```

```

]
 },
 {
 id: 9, name: 'Web Team',
 child: [
 { id: 11, pid: 9, name: 'Joshua' },
 { id: 12, pid: 9, name: 'Matthew' },
 { id: 13, pid: 9, name: 'David' },
]
 },
 {
 id: 14, name: 'Build Team',
 child: [
 { id: 15, pid: 14, name: 'Ryan' },
 { id: 16, pid: 14, name: 'Justin' },
 { id: 17, pid: 14, name: 'Robert' },
]
 },
 {
 id: 18, name: 'WPF Team',
 child: [
 { id: 19, pid: 18, name: 'Brown' },
 { id: 20, pid: 18, name: 'Johnson' },
 { id: 21, pid: 18, name: 'Miller' },
]
 }
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: productTeam, id: 'id', text: 'name', child:
'child', selected: 'isSelected' },
 allowDragAndDrop: true,
});
treeObj.appendTo('#tree');

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

<div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Multiple-node drag and drop

To drag and drop more than one node, you should enable the [allowMultiSelection](#) property along with the [allowDragAndDrop](#) property.

To perform multi-selection, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following sample, the [allowMultiSelection](#) property is enabled along with the [allowDragAndDrop](#) property.

### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
let productTeam: { [key: string]: Object }[] = [
 {
 id: 1, name: 'ASP.NET MVC Team', expanded: true,
 child: [
 { id: 2, pid: 1, name: 'Smith' },
 { id: 3, pid: 1, name: 'Johnson', isSelected: true },
 { id: 4, pid: 1, name: 'Anderson', isSelected: true },
]
 },
 {
 id: 5, name: 'Windows Team',
 child: [
 { id: 6, pid: 5, name: 'Clark' },
 { id: 7, pid: 5, name: 'Wright' },
 { id: 8, pid: 5, name: 'Lopez' },
]
 },
 {
 id: 9, name: 'Web Team',
 child: [
 { id: 11, pid: 9, name: 'Joshua' },
 { id: 12, pid: 9, name: 'Matthew' },
 { id: 13, pid: 9, name: 'David' },
]
 },
 {
 id: 14, name: 'Build Team',

```

```

 child: [
 { id: 15, pid: 14, name: 'Ryan' },
 { id: 16, pid: 14, name: 'Justin' },
 { id: 17, pid: 14, name: 'Robert' },
]
 },
 {
 id: 18, name: 'WPF Team',
 child: [
 { id: 19, pid: 18, name: 'Brown' },
 { id: 20, pid: 18, name: 'Johnson' },
 { id: 21, pid: 18, name: 'Miller' },
]
 }
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: productTeam, id: 'id', text: 'name', child:
'child', selected: 'isSelected' },
 allowDragAndDrop: true,
 allowMultiSelection: true,
});
treeObj.appendTo('#tree');

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>

```

```
<script src="index.js" type="text/javascript"></script>
</body></html>
```

See Also

- [How to restrict the drag-and-drop for particular tree nodes](#)

### Template in EJ2 JavaScript Treeview control

The TreeView component allows you to customize the look of TreeView nodes by using the [nodeTemplate](#) property. This property accepts either [template string](#) or HTML element ID.

In the following sample, employee information such as employee photo, name, and designation have been included using the `nodeTemplate` property.

The template expression should be provided inside the `${...}` interpolation syntax.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
let employees: { [key: string]: Object }[] = [
 { id: 1, name: 'Steven Buchanan', eimg: '10', job: 'CEO', hasChild:
true, expanded: true },
 { id: 2, pid: 1, name: 'Laura Callahan', eimg: '2', job: 'Product
Manager', hasChild: true },
 { id: 3, pid: 2, name: 'Andrew Fuller', eimg: '7', job: 'Team Lead',
hasChild: true },
 { id: 4, pid: 3, name: 'Anne Dodsworth', eimg: '1', job: 'Developer' },
 { id: 5, pid: 1, name: 'Nancy Davolio', eimg: '4', job: 'Product
Manager', hasChild: true },
 { id: 6, pid: 5, name: 'Michael Suyama', eimg: '9', job: 'Team Lead',
hasChild: true },
 { id: 7, pid: 6, name: 'Robert King', eimg: '8', job: 'Developer' },
 { id: 8, pid: 7, name: 'Margaret Peacock', eimg: '6', job: 'Developer'
},
 { id: 9, pid: 1, name: 'Janet Leverling', eimg: '3', job: 'HR' },
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: employees, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' },
 cssClass: 'custom',
 nodeTemplate: '#treeTemplate'
});
treeObj.appendTo('#tree');
```

#### INDEX.HTML

```
<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
```

```

<title>Essential JS 2 for TreeView </title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 for TreeView UI
Control">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
</div>
<script id="treeTemplate" type="text/x-template">
 <div>
 <img class="eimage" src=
"https://ej2.syncfusion.com/demos/src/treeview/images/employees/${eimg}.png"
alt="${eimg}" />
 <div class="ename">${name}</div>
 <div class="ejob">${job}</div>
 </div>
</script>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## INDEX.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 width: 30%;
 position: absolute;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 top: 45%;
 left: 45%;
}
#treeparent {
 display: block;
 max-width: 450px;
 max-height: 350px;
}

```

```

margin: auto;
overflow: auto;
border: 1px solid #dddddd;
border-radius: 3px;
}
.custom .e-list-item .e-fullrow {
 height: 72px;
}
.custom .e-list-item .e-list-text {
 line-height: normal;
}
.eimage {
 float: left;
 padding: 11px 16px 11px 0;
}
.ename {
 font-size: 16px;
 padding: 14px 0 0;
}
.ejob {
 font-size: 14px;
 opacity: .87;
}

```

### See Also

- [How to customize the expand and collapse icons](#)
- [How to customize the tree nodes based on levels](#)

### Accessibility in EJ2 JavaScript Treeview component

The TreeView component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the TreeView component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### WAI-ARIA attributes

The TreeView component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the TreeView component:

| Attributes | Purpose |

| --- | --- |

| **role=tree** | All tree nodes are contained within the element. |

| **role=treeitem** | Specifies the role of each tree node in a selectable TreeView and its containment within the tree. |

| **role=group** | Specifies the role of each parent node container. |

| **role=checkbox** | Indicates checkbox control along with treeitem element. |

| **aria-multiselectable** | Indicates whether the TreeView enables multiple selection or not. |

| **aria-expanded** | Indicates whether the parent node has expanded or not. |

| **aria-selected** | Indicates the selected node. |

| **aria-grabbed** | Indicates the selected state on drag-and-drop of node. |

| **aria-level** | Indicates the level of node in TreeView. |

| **aria-checked** | Indicates the current checked state of TreeView checkbox. |



- | `aria-label` | Indicates the contextual message for the TreeView checkbox. |
- | `aria-activedescendant` | Identifies the currently active element when focusing on the TreeView. |
- | `aria-disabled` | Indicates element is perceivable but disabled. |

### Keyboard interaction

The TreeView component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the TreeView component.

Interaction Keys	Description
----- -----	
Arrow Up	Goes to the previous node.
Arrow Down	Goes to the next node.
Arrow Right	Expands the current node.
Arrow Left	Collapses the current node.
Home	Goes to the first node.
End	Goes to the last node.
F2	Edits the focused node.
Esc	Focuses out the edit state without saving the edited text.
Enter	Selects the focused node/saves the edited text.
Space	Checks the current node.
Ctrl + A	Selects all nodes.

### Ensuring accessibility

The TreeView component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the TreeView component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the TreeView component with accessibility tools.

See also

- [Accessibility in Syncfusion EJ2 JavaScript components](#)

### How To

Customize the expand and collapse icons in EJ2 JavaScript Treeview control

You can customize TreeView expand and collapse icons by using the `cssClass` property of TreeView.

Refer to the sample to customize expand/collapse icons.

### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { TreeView } from '@syncfusion/ej2-navigations';
let hierarchicalData: { [key: string]: Object }[] = [
 { id: '01', name: 'Local Disk (C:)', expanded: true,
 subChild: [
 {
 id: '01-01', name: 'Program Files',
 subChild: [
 { id: '01-01-01', name: '7-Zip' },
 { id: '01-01-02', name: 'Git' },
 { id: '01-01-03', name: 'IIS Express' },
]
 },
 {
 id: '01-02', name: 'Users', expanded: true,
 subChild: [
 { id: '01-02-01', name: 'Smith' },
 { id: '01-02-02', name: 'Public' },
 { id: '01-02-03', name: 'Admin' },
]
 },
 {
 id: '01-03', name: 'Windows',
 subChild: [
 { id: '01-03-01', name: 'Boot' },
 { id: '01-03-02', name: 'FileManager' },
 { id: '01-03-03', name: 'System32' },
]
 }
]
 },
 {
 id: '02', name: 'Local Disk (D:)',
 subChild: [
 {
 id: '02-01', name: 'Personals',
 subChild: [
 { id: '02-01-01', name: 'My photo.png' },
 { id: '02-01-02', name: 'Rental document.docx' },
 { id: '02-01-03', name: 'Pay slip.pdf' },
]
 },
 {
 id: '02-02', name: 'Projects',
 subChild: [
 { id: '02-02-01', name: 'ASP Application' },
 { id: '02-02-02', name: 'TypeScript Application' },
 { id: '02-02-03', name: 'React Application' },
]
 },
 {
 id: '02-03', name: 'Office',
 subChild: [
 { id: '02-03-01', name: 'Work details.docx' },
 { id: '02-03-02', name: 'Weekly report.docx' },
 { id: '02-03-03', name: 'Wish list.csv' },
]
 }
]
 }
]

```

```

],
 },
],
},
{
 id: '03', name: 'Local Disk (E:)', icon: 'folder',
 subChild: [
 {
 id: '03-01', name: 'Pictures',
 subChild: [
 { id: '03-01-01', name: 'Wind.jpg' },
 { id: '03-01-02', name: 'Stone.jpg' },
 { id: '03-01-03', name: 'Home.jpg' },
]
 },
 {
 id: '03-02', name: 'Documents',
 subChild: [
 { id: '03-02-01', name: 'Environment Pollution.docx' },
 { id: '03-02-02', name: 'Global Warming.ppt' },
 { id: '03-02-03', name: 'Social Network.pdf' },
]
 },
 {
 id: '03-03', name: 'Study Materials',
 subChild: [
 { id: '03-03-01', name: 'UI-Guide.pdf' },
 { id: '03-03-02', name: 'Tutorials.zip' },
 { id: '03-03-03', name: 'TypeScript.7z' },
]
 }
],
}
];
let treeObj: TreeView = new TreeView({
 fields: { dataSource: hierarchicalData, id: 'id', text: 'name', child:
'subChild' },
 cssClass: 'custom'
});
treeObj.appendTo('#tree');

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## INDEX.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 width: 30%;
 position: absolute;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 top: 45%;
 left: 45%;
}
#treeparent {
 display: block;
 max-width: 450px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
}
.custom .e-list-item .e-icons {
 font-family: "Customize-icon";
}
.custom.e-treeview .e-list-item .e-icon-expandable::before, .custom.e-
treeview .e-list-item .e-icon-collapsible::before {
 content: '\e700';
 font-size: 12px;
}
@font-face {
 font-family: 'Customize-icon';

```

[illegible]

## Process the tree node operations using context menu in EJ2 JavaScript Treeview control

You can intergrade the context menu with 'TreeView' component in order to perform the tree view related operations like add, remove and renaming node.

Following is an example which demonstrates the above cases which are used to manipulate tree view operations in the 'select ' event of context menu.

## INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
import { ContextMenu, MenuEventArgs, MenuItemModel, ContextMenuModel,
TreeView, BeforeOpenCloseMenuEventArgs, NodeClickEventArgs } from
 '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * The node operation in tree view using context menu
 */
//define the data source for TreeView
let data: { [key: string]: Object } [] = [
 { id: '1', name: 'Local Disk (C:)', hasAttribute: { class: 'remove
rename' }, hasChild: true, expanded: true, },
 { id: '2', name: 'Program Files', pid: '1', hasChild: true },
 { id: '3', name: 'Windows NT', pid: '2' },
 { id: '4', name: 'Windows Mail', pid: '2' },
 { id: '5', name: 'Windows Photo Viewer', pid: '2' },
```

```

 { id: '6', name: 'Users', pid: '1', hasChild: true, expanded: true },
 { id: '7', name: 'Smith', pid: '6' },
 { id: '8', name: 'Public', pid: '6' },
 { id: '9', name: 'Admin', pid: '6' },
 { id: '10', name: 'Windows', pid: '1', hasChild: true },
 { id: '11', name: 'Boot', pid: '10' },
 { id: '12', name: 'FileManager', pid: '10' },
 { id: '13', name: 'System32', pid: '10' },
 { id: '14', name: 'Local Disk (D:)', hasAttribute: { class: 'remove' },
hasChild: true},
 { id: '15', name: 'Personals', pid: '14', hasChild: true },
 { id: '16', name: 'My photo.png', pid: '15' },
 { id: '17', name: 'Rental document.docx', pid: '15' },
 { id: '18', name: 'Pay slip.pdf', pid: '15' },
 { id: '19', name: 'Projects', pid: '14', hasChild: true },
 { id: '20', name: 'ASP Application', pid: '19' },
 { id: '21', name: 'TypeScript Application', pid: '19' },
 { id: '22', name: 'React Application', pid: '19' },
 { id: '23', name: 'Office', pid: '14', hasChild: true },
 { id: '24', name: 'Work details.docx', pid: '23' },
 { id: '25', name: 'Weekly report.docx', pid: '23' },
 { id: '26', name: 'Wish list.csv', pid: '23' },
 { id: '27', name: 'Local Disk (E:)', hasAttribute: { class: 'remove' },
hasChild: true },
 { id: '28', name: 'Pictures', pid: '27', hasChild: true },
 { id: '29', name: 'Wind.jpg', pid: '28' },
 { id: '30', name: 'Stone.jpg', pid: '28' },
 { id: '31', name: 'Home.jpg', pid: '28' },
 { id: '32', name: 'Documents', pid: '27', hasChild: true },
 { id: '33', name: 'Environment Pollution.docx', pid: '32' },
 { id: '34', name: 'Global Warming.ppt', pid: '32' },
 { id: '35', name: 'Social Network.pdf', pid: '32' },
 { id: '36', name: 'Study Materials', pid: '27', hasChild: true },
 { id: '37', name: 'UI-Guide.pdf', pid: '36' },
 { id: '38', name: 'Tutorials.zip', pid: '36' },
 { id: '39', name: 'TypeScript.7z', pid: '36' }
];
// Render the TreeView by mapping its fields property with data source
properties
let treeObj: TreeView = new TreeView({
 fields: { dataSource: data, id: 'id', text: 'name', parentID: 'pid',
hasChildren: 'hasChild', htmlAttributes: 'hasAttribute'},
 nodeClicked: nodeClick
});
treeObj.appendTo('#tree');
function nodeClick(args: NodeClickEventArgs) {
 if (args.event.which === 3) {
 treeObj.selectedNodes = [args.node.getAttribute('data-uid')]
 }
}
//Render the context menu with target as Treeview
let menuItems: MenuItemModel[] = [
 { text: 'Add New Item' },
 { text: 'Rename Item' },
 { text: 'Remove Item' }
];
let menuOptions: ContextMenuModel = {

```

```

 target: '#tree',
 items: menuItems,
 select: menuclick,
 beforeOpen: beforeopen
 };
 let menuObj: ContextMenu = new ContextMenu(menuOptions, '#contextmenu');
 let index: number = 1;
 function menuclick(args: MenuEventArgs) {
 let targetNodeId: string = treeObj.selectedNodes[0];
 if (args.item.text == "Add New Item") {
 let nodeId: string = "tree_" + index;
 let item: { [key: string]: Object } = { id: nodeId, name: "New Folder"
};
 treeObj.addNodes([item], targetNodeId, null);
 index++;
 data.push(item);
 treeObj.beginEdit(nodeId);
 }
 else if (args.item.text == "Remove Item") {
 treeObj.removeNodes([targetNodeId]);
 }
 else if (args.item.text == "Rename Item") {
 treeObj.beginEdit(targetNodeId);
 }
 }
 }
 function beforeopen(args: BeforeOpenCloseMenuEventArgs) {
 let targetNodeId: string = treeObj.selectedNodes[0];
 let targetNode: Element = document.querySelector('[data-uid="' +
targetNodeId + '"]');
 if (targetNode.classList.contains('remove')) {
 menuObj.enableItems(['Remove Item'], false);
 }
 else {
 menuObj.enableItems(['Remove Item'], true);
 }
 if (targetNode.classList.contains('rename')) {
 menuObj.enableItems(['Rename Item'], false);
 }
 else {
 menuObj.enableItems(['Rename Item'], true);
 }
 }
}

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 <ul id="contextmenu">
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Check uncheck the checkbox on clicking the tree node text in EJ2 JavaScript Treeview control

You can check and uncheck the checkboxes of tree view by clicking the tree node using the `nodeClicked` event of TreeView.

### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * TreeView check/uncheck the check box, while clicking on the tree node
 * text sample
 */
// Data source for TreeView component
let countries: { [key: string]: Object } [] = [
 { id: 1, name: 'Australia', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'New South Wales' },
 { id: 3, pid: 1, name: 'Victoria' },
 { id: 4, pid: 1, name: 'South Australia' },
 { id: 6, pid: 1, name: 'Western Australia' },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'China', hasChild: true },
 { id: 12, pid: 11, name: 'Guangzhou' },
 { id: 13, pid: 11, name: 'Shanghai' },
 { id: 14, pid: 11, name: 'Beijing' },
 { id: 15, pid: 11, name: 'Shantou' },
 { id: 16, name: 'France', hasChild: true },

```



```

 { id: 17, pid: 16, name: 'Pays de la Loire' },
 { id: 18, pid: 16, name: 'Aquitaine' },
 { id: 19, pid: 16, name: 'Brittany' },
 { id: 20, pid: 16, name: 'Lorraine' },
 { id: 21, name: 'India', hasChild: true },
 { id: 22, pid: 21, name: 'Assam' },
 { id: 23, pid: 21, name: 'Bihar' },
 { id: 24, pid: 21, name: 'Tamil Nadu' },
 { id: 25, pid: 21, name: 'Punjab' }
];

 // Render the TreeView with checkboxes
 let treeObj: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' },
 showCheckBox: true,
 nodeClicked: nodeCheck,
 keyPress: nodeCheck
 });
 treeObj.appendTo('#tree');
 function nodeCheck(args: NodeKeyPressEventArgs | NodeClickEventArgs): void
 {
 let checkedNode: any = [args.node];
 if (args.event.target.classList.contains('e-fullrow') || args.event.key
== "Enter") {
 let getNodeDetails: any = treeObj.getNode(args.node);
 if (getNodeDetails.isChecked == 'true') {
 treeObj.uncheckAll(checkedNode);
 } else {
 treeObj.checkAll(checkedNode);
 }
 }
 }
}

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

```

```

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Validate the text when renaming the tree node in EJ2 JavaScript Treeview control

You can validate the tree node text while editing using `nodeEdited` event of the TreeView. Following is an example that shows how to validate and prevent empty values in tree node.

#### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * TreeView node editing sample with validation
 */
// Hierarchical data source for TreeView component
let treeData: { [key: string]: Object } [] = [
 {
 id: 1, name: 'Discover Music', expanded: true,
 child: [
 { id: 2, name: 'Hot Singles' },
 { id: 3, name: 'Rising Artists' },
 { id: 4, name: 'Live Music' }
]
 },
 {
 id: 7, name: 'Sales and Events',
 child: [
 { id: 8, name: '100 Albums - $5 Each' },
 { id: 9, name: 'Hip-Hop and R&B Sale' },
 { id: 10, name: 'CD Deals' }
]
 },
 {
 id: 11, name: 'Categories',
 child: [
 { id: 12, name: 'Songs' },
 { id: 13, name: 'Bestselling Albums' },
 { id: 14, name: 'New Releases' },
 { id: 15, name: 'Bestselling Songs' }
]
 },
 {
 id: 16, name: 'MP3 Albums',

```

```

 child: [
 { id: 17, name: 'Rock' },
 { id: 18, name: 'Gospel' },
 { id: 19, name: 'Latin Music' },
 { id: 20, name: 'Jazz' }
],
 },
 {
 id: 21, name: 'More in Music',
 child: [
 { id: 22, name: 'Music Trade-In' },
 { id: 23, name: 'Redeem a Gift Card' },
 { id: 24, name: 'Band T-Shirts' }
]
 }
];

// Render the TreeView with editing option
let treeObj: TreeView = new TreeView({
 fields: { dataSource: treeData, id: 'id', text: 'name', child: 'child'
},
 allowEditing: true,
 nodeEdited: onNodeEdited
});
treeObj.appendTo('#tree');
function onNodeEdited(args: NodeEditEventArgs): void {
 let displayContent:string = "";
 if (args.newText.trim() == "") {
 args.cancel=true;
 displayContent = "TreeView item text should not be empty";
 }
 else if (args.newText != args.oldText) {
 displayContent = "TreeView item text edited successfully";
 } else {
 displayContent = "";
 }
 document.getElementById("display").innerHTML = displayContent;
}

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 <div id="display"></div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Customize the tree nodes based on levels in EJ2 JavaScript Treeview control

You can customize the tree nodes level wise by adding custom cssClass to the component and enabling styles.

#### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * TreeView customize the tree nodes in level wise
 */
// Hierarchical data source for TreeView component
let hierarchicalData: { [key: string]: Object } [] = [
{
 id: '01', name: 'Local Disk (C:)', expanded: true,
 subChild: [
 {
 id: '01-01', name: 'Program Files',
 subChild: [
 { id: '01-01-01', name: 'Windows NT' },
 { id: '01-01-02', name: 'Windows Mail' },
 { id: '01-01-03', name: 'Windows Photo Viewer' },
]
 },
 {
 id: '01-02', name: 'Users', expanded: true,
 subChild: [
 { id: '01-02-01', name: 'Smith' },
 { id: '01-02-02', name: 'Public' },
 { id: '01-02-03', name: 'Admin' },
]
 },
],
}
]

```

```

 id: '01-03', name: 'Windows',
 subChild: [
 { id: '01-03-01', name: 'Boot' },
 { id: '01-03-02', name: 'FileManager' },
 { id: '01-03-03', name: 'System32' },
]
 },
]
},
{
 id: '02', name: 'Local Disk (D:)',
 subChild: [
 {
 id: '02-01', name: 'Personals',
 subChild: [
 { id: '02-01-01', name: 'My photo.png' },
 { id: '02-01-02', name: 'Rental document.docx' },
 { id: '02-01-03', name: 'Pay slip.pdf' },
]
 },
 {
 id: '02-02', name: 'Projects',
 subChild: [
 { id: '02-02-01', name: 'ASP Application' },
 { id: '02-02-02', name: 'TypeScript Application' },
 { id: '02-02-03', name: 'React Application' },
]
 },
 {
 id: '02-03', name: 'Office',
 subChild: [
 { id: '02-03-01', name: 'Work details.docx' },
 { id: '02-03-02', name: 'Weekly report.docx' },
 { id: '02-03-03', name: 'Wish list.csv' },
]
 },
]
},
{
 id: '03', name: 'Local Disk (E:)', icon: 'folder',
 subChild: [
 {
 id: '03-01', name: 'Pictures',
 subChild: [
 { id: '03-01-01', name: 'Wind.jpg' },
 { id: '03-01-02', name: 'Stone.jpg' },
 { id: '03-01-03', name: 'Home.jpg' },
]
 },
 {
 id: '03-02', name: 'Documents',
 subChild: [
 { id: '03-02-01', name: 'Environment Pollution.docx' },
 { id: '03-02-02', name: 'Global Warming.ppt' },
 { id: '03-02-03', name: 'Social Network.pdf' },
]
 },
]
},

```

```

 {
 id: '03-03', name: 'Study Materials',
 subChild: [
 { id: '03-03-01', name: 'UI-Guide.pdf' },
 { id: '03-03-02', name: 'Tutorials.zip' },
 { id: '03-03-03', name: 'TypeScript.7z' },
]
 },
],
}
];

// Render the TreeView by mapping its fields property with data source
properties
let treeObj: TreeView = new TreeView({
 fields: { dataSource: hierarchicalData, id: 'id', text: 'name', child:
'subChild' },
 cssClass: ("mytree")
});
treeObj.appendTo('#tree');
```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 <div class="details">
 <label>Note:</label>
 <div>1. The font-weight "Bold" is applied for all the
leaf nodes</div>
 <div><i>2. The font-weight "Italic" is applied for first
level nodes</i></div>
 <div style="color: darkmagenta">3. The color "darkmagenta"
is applied for second level nodes</div>
 </div>
 </div>
 </div>
</div>
```

```

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### **INDEX.CSS**

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 width: 30%;
 position: absolute;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 top: 45%;
 left: 45%;
}
#treeparent {
 display: block;
 max-width: 350px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
}
.details {
 padding-left: 10px;
}
/*apply custom css to first level*/
.mytree .e-level-1 > .e-text-content .e-list-text {
 font-style: italic;
}
/*apply custom css to second level*/
.mytree .e-level-2 > .e-text-content .e-list-text {
 color: darkmagenta;
}
/*apply custom css to all the leaf nodes*/
.mytree .e-level-3 > .e-text-content .e-list-text {
 font-weight: bold;
}

```

### Restrict the drag and drop for particular tree nodes in EJ2 JavaScript Treeview control

You can able to restrict to drag and drop files under folder only. These can be achieved by using 'nodeDragStop' and 'nodeDragging' event of TreeView.

### **INDEX.TS**

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
// Hierarchical data source for TreeView component
let hierarchicalData: { [key: string]: Object }[] = [
 {
 nodeId: '01', nodeText: 'Music', icon: 'folder',
 nodeChild: [
 { nodeId: '01-01', nodeText: 'Gouttes.mp3', icon: 'audio' }
]
 },
 {
 nodeId: '02', nodeText: 'Videos', icon: 'folder',
 nodeChild: [
 { nodeId: '02-01', nodeText: 'Naturals.mp4', icon: 'video' },
 { nodeId: '02-02', nodeText: 'Wild.mpeg', icon: 'video' },
]
 },
 {
 nodeId: '03', nodeText: 'Documents', icon: 'folder',
 nodeChild: [
 { nodeId: '03-01', nodeText: 'Environment Pollution.docx', icon: 'docx' },
 { nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx', icon: 'docx' },
 { nodeId: '03-03', nodeText: 'Global Warming.ppt', icon: 'ppt' },
 { nodeId: '03-04', nodeText: 'Social Network.pdf', icon: 'pdf' },
 { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf', icon: 'pdf' },
]
 },
 {
 nodeId: '04', nodeText: 'Pictures', icon: 'folder', expanded: true,
 nodeChild: [
 {
 nodeId: '04-01', nodeText: 'Camera Roll', icon: 'folder', expanded: true,
 nodeChild: [
 { nodeId: '04-01-01', nodeText: 'WIN_20160726_094117.JPG', image: 'https://ej2.syncfusion.com/demos/src/treeview/images/employees/9.png' },
 { nodeId: '04-01-02', nodeText: 'WIN_20160726_094118.JPG', image: 'https://ej2.syncfusion.com/demos/src/treeview/images/employees/3.png' },
]
 },
 { nodeId: '04-02', nodeText: 'Wind.jpg', icon: 'images' },
 { nodeId: '04-03', nodeText: 'Stone.jpg', icon: 'images' },
]
 },
 {
 nodeId: '05', nodeText: 'Downloads', icon: 'folder',
 nodeChild: [

```



```

 { nodeId: '05-01', nodeText: 'UI-Guide.pdf', icon: 'pdf' },
 { nodeId: '05-02', nodeText: 'Tutorials.zip', icon: 'zip' },
 { nodeId: '05-03', nodeText: 'Game.exe', icon: 'exe' },
 { nodeId: '05-04', nodeText: 'TypeScript.7z', icon: 'zip' },
],
},
];
// Render the TreeView with image icons
let treeObj: TreeView = new TreeView({
 fields: { dataSource: hierarchicalData, id: 'nodeId', text:
'nodeText', child: 'nodeChild', iconCss: 'icon', imageUrl: 'image' },
 sortOrder: 'Ascending',
 allowDragAndDrop: true,
 nodeDragStop: dragStop,
 nodeDragging: nodeDrag
});
treeObj.appendTo('#tree');
function nodeDrag(args: DragAndDropEventArgs): void {
 if (args.droppedNode != null &&
args.droppedNode.getElementsByClassName('folder') &&
args.droppedNode.getElementsByClassName('folder').length === 0) {
 args.dropIndicator = 'e-no-drop';
 }
}
function dragStop(args: DragAndDropEventArgs): void {
 if (args.droppedNode != null &&
args.droppedNode.getElementsByClassName('folder') &&
args.droppedNode.getElementsByClassName('folder').length === 0) {
 args.cancel = true;
 }
}
}

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <div id="container">
 <div id="treeparent">

```

```
<div id="tree"></div>
</div>
<style>
.e-treeview .e-list-img {
 width: 25px;
 height: 25px;
}
/* Loading sprite image for TreeView */
.e-treeview .e-list-icon {
 background-repeat: no-repeat;
 background-image:
url(https://ej2.syncfusion.com/demos/src/treeview/images/icons/file_icons.png);
 height: 20px;
}
/* Specify the icon positions based upon class name */
.e-treeview .e-list-icon.folder {
 background-position: -10px -552px;
}
.e-treeview .e-list-icon.docx {
 background-position: -10px -20px;
}
.e-treeview .e-list-icon.ppt {
 background-position: -10px -48px;
}
.e-treeview .e-list-icon.pdf {
 background-position: -10px -104px;
}
.e-treeview .e-list-icon.images {
 background-position: -10px -132px;
}
.e-treeview .e-list-icon.zip {
 background-position: -10px -188px;
}
.e-treeview .e-list-icon.audio {
 background-position: -10px -244px;
}
.e-treeview .e-list-icon.video {
 background-position: -10px -272px;
}
.e-treeview .e-list-icon.exe {
 background-position: -10px -412px;
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Accordion tree in EJ2 JavaScript Treeview control

Accordion is an interface where a list of items can be collapsed or expanded, but only one list can be collapsed or expanded at a time. You can customize the TreeView to make it behave as an accordion. Refer to the following code sample to create an accordion tree.

#### INDEX.TS

```
import { TreeView } from '@syncfusion/ej2-navigations';
/**
 * Accordion tree sample
 */
// Hierarchical data source for TreeView component
let continents: { [key: string]: Object; }[] = [
 {
 code: "AF", name: "Africa", countries: [
 { code: "NGA", name: "Nigeria" },
 { code: "EGY", name: "Egypt" },
 { code: "ZAF", name: "South Africa" }
]
 },
 {
 code: "AS", name: "Asia", countries: [
 { code: "CHN", name: "China" },
 { code: "IND", name: "India", selected: true },
 { code: "JPN", name: "Japan" }
]
 },
 {
 code: "EU", name: "Europe", countries: [
 { code: "DNK", name: "Denmark" },
 { code: "FIN", name: "Finland" },
 { code: "AUT", name: "Austria",
 }
]
 },
 {
 code: "NA", name: "North America", countries: [
 { code: "USA", name: "United States of America" },
 { code: "CUB", name: "Cuba" },
 { code: "MEX", name: "Mexico" }
]
 },
 {
 code: "OC", name: "Oceania", countries: [
 { code: "AUS", name: "Australia" },
 { code: "NZL", name: "New Zealand" },
 { code: "WSM", name: "Samoa" }
]
 },
];
// Render treeview with cssClass
let tree1: TreeView = new TreeView({
 fields: { dataSource: continents, id: "code", text: "name", child:
"countries" },
 nodeSelected: nodeSelect,
 cssClass: ("accordiontree")
});
```

```
tree1.appendTo('#tree');
function nodeSelect (args: NodeSelectEventArgs) {
 if (args.node.classList.contains('e-level-1')) {
 this.collapseAll();
 this.expandAll([args.node]);
 }
}
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## INDEX.CSS

```
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 width: 30%;
```

```

position: absolute;
font-family: 'Helvetica Neue','calibri';
font-size: 14px;
top: 45%;
left: 45%;
}
#treeparent {
 display: block;
 max-width: 350px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 padding-top: 50px;
}
.accordiontree .e-list-item.e-level-1 > .e-fullrow, .accordiontree .e-list-item.e-level-1.e-active > .e-fullrow, .accordiontree .e-list-item.e-level-1.e-hover > .e-fullrow, .accordiontree .e-list-item.e-level-1 > .e-fullrow, .accordiontree .e-list-item.e-level-1.e-active.e-hover > .e-fullrow {
 background-color: darkslateblue;
 border-color: darkslateblue;
}
.accordiontree .e-list-item.e-level-1 > .e-text-content .e-list-text, .accordiontree .e-list-item.e-level-1.e-active > .e-text-content .e-list-text, .accordiontree .e-list-item.e-level-1.e-hover > .e-text-content .e-list-text, .accordiontree .e-list-item.e-level-1.e-active.e-hover > .e-text-content .e-list-text {
 color: white;
 font-size: 16px;
}
.accordiontree .e-list-item.e-level- .e-icons.e-icon-collapsible, .accordiontree .e-list-item.e-level-1 .e-icons.e-icon-collapsible, .accordiontree .e-list-item.e-level-1 .e-icon-expandable {
 display: none
}
.accordiontree .e-list-item.e-level-2 > .e-fullrow, .accordiontree .e-list-item.e-level-2.e-active > .e-fullrow, .accordiontree .e-list-item.e-level-2.e-hover > .e-fullrow, .accordiontree .e-list-item.e-level-2 > .e-fullrow, .accordiontree .e-list-item.e-level-2.e-active.e-hover > .e-fullrow {
 background-color: white;
 border-color: white;
}
.accordiontree .e-list-item.e-level-2 > .e-text-content .e-list-text, .accordiontree .e-list-item.e-level-2.e-active > .e-text-content .e-list-text, .accordiontree .e-list-item.e-level-2.e-hover > .e-text-content .e-list-text, .accordiontree .e-list-item.e-level-2.e-active.e-hover > .e-text-content .e-list-text {
 color: blue;
 font-size: 14px;
}
}

```

### Auto hide show expand collapse icon in EJ2 JavaScript Treeview control

You can display the expand icon by hovering the mouse over TreeView and hide the expand icon by leaving the mouse from TreeView. Refer to the following code sample to hide/show the expand/collapse icon automatically using the mouse.

**INDEX.TS**

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * TreeView Auto hide/show expand/collapse icons
 */
// List data source for TreeView component
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'India', hasChild: true },
 { id: 2, pid: 1, name: 'Assam' },
 { id: 3, pid: 1, name: 'Bihar' },
 { id: 4, pid: 1, name: 'Tamil Nadu' },
 { id: 6, pid: 1, name: 'Punjab' },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'France', hasChild: true },
 { id: 12, pid: 11, name: 'Pays de la Loire' },
 { id: 13, pid: 11, name: 'Aquitaine' },
 { id: 14, pid: 11, name: 'Brittany' },
 { id: 15, pid: 11, name: 'Lorraine' },
 { id: 16, name: 'Australia', hasChild: true },
 { id: 17, pid: 16, name: 'New South Wales' },
 { id: 18, pid: 16, name: 'Victoria' },
 { id: 19, pid: 16, name: 'South Australia' },
 { id: 20, pid: 16, name: 'Western Australia' },
 { id: 21, name: 'China', hasChild: true },
 { id: 22, pid: 21, name: 'Guangzhou' },
 { id: 23, pid: 21, name: 'Shanghai' },
 { id: 24, pid: 21, name: 'Beijing' },
 { id: 25, pid: 21, name: 'Shantou' }
];
// Renders treeview
let tree1: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', text: 'name', parentID:
'pid', hasChildren: 'hasChild' },
 created: onCreate
});
tree1.appendTo('#tree');
function onCreate() {
 let treeElement: Element = document.getElementById("tree");
 let collapse: NodeListOf<Element> = treeElement.querySelectorAll('.e-
icons.e-icon-collapsible');
 let expand: NodeListOf<Element> = treeElement.querySelectorAll('.e-
icons.e-icon-expandable');
 hideIcon(expand, collapse);
 document.getElementById("tree").addEventListener('mouseenter', (event:any)
=> {
 showIcon(expand, collapse);
 })
 document.getElementById("tree").addEventListener('mouseleave', (event:any)
=> {
 hideIcon(expand, collapse);
 })
}

```

```

}
// hides expand/collapse icon on hovering the mouse
function hideIcon(expand: NodeListOf<Element>, collapse:
NodeListOf<Element>) {
 for(let i: number = 0; i < collapse.length; i++){
 collapse[i].setAttribute('style','visibility: hidden');
 }
 for(let j: number = 0; j < expand.length; j++){
 expand[j].setAttribute('style','visibility: hidden');
 }
}
// shows expand/collapse icon while leaving the mouse
function showIcon(expand: NodeListOf<Element>, collapse:
NodeListOf<Element>) {
 for(let i: number = 0; i < collapse.length; i++){
 collapse[i].setAttribute('style',"visibility", "");
 }
 for(let j: number = 0; j < expand.length; j++){
 expand[j].setAttribute('style',"visibility", "");
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";

```

```

}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Filtering tree nodes in EJ2 JavaScript Treeview control

You can filter the tree nodes based on their text using the **DataManager** plugin and the **fields** property of the TreeView.

The following code example demonstrates how to filter the tree nodes in a TreeView.

#### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
import { MaskedTextBox } from '@syncfusion/ej2-inputs';
import { DataManager, Query, Predicate } from '@syncfusion/ej2-data';
enableRipple(true);

// local data source for TreeView component
let localData: { [key: string]: Object }[] = [
 { id: 1, name: "Australia", hasChild: true },
 { id: 2, pid: 1, name: "New South Wales" },
 { id: 3, pid: 1, name: "Victoria" },
 { id: 4, pid: 1, name: "South Australia" },
 { id: 6, pid: 1, name: "Western Australia" },
 { id: 7, name: "Brazil", hasChild: true },
 { id: 8, pid: 7, name: "Paraná" },
 { id: 9, pid: 7, name: "Ceará" },
 { id: 10, pid: 7, name: "Acre" },
 { id: 11, name: "China", hasChild: true },
 { id: 12, pid: 11, name: "Guangzhou" },
 { id: 13, pid: 11, name: "Shanghai" },
 { id: 14, pid: 11, name: "Beijing" },
 { id: 15, pid: 11, name: "Shantou" },
 { id: 16, name: "France", hasChild: true },
 { id: 17, pid: 16, name: "Pays de la Loire" },
 { id: 18, pid: 16, name: "Aquitaine" },
 { id: 19, pid: 16, name: "Brittany" },
 { id: 20, pid: 16, name: "Lorraine" },
 { id: 21, name: "India", hasChild: true },
 { id: 22, pid: 21, name: "Assam" },
 { id: 23, pid: 21, name: "Bihar" },
 { id: 24, pid: 21, name: "Tamil Nadu" },
 { id: 25, pid: 21, name: "Punjab" }
];

// Render the TreeView with image icons
let listTreeObj: TreeView = new TreeView({
 fields: { dataSource: localData, id: 'id', parentID: 'pid', text:
'name', hasChildren: 'hasChild' },
});
listTreeObj.appendTo('#treeView');
let mask: MaskedTextBox = new MaskedTextBox({
 placeholder: "Enter the tree node to search",
 change: searchNodes
});
mask.appendTo('#mask');

```



```

//Change the dataSource for TreeView
function changeDataSource(data) {
 listTreeObj.fields = {
 dataSource: data, id: 'id', text: 'name',
 parentID: 'pid', hasChildren: 'hasChild'
 }
}

//Filtering the TreeNodes
function searchNodes(args) {
 let _text = mask.element.value;
 let predicats = [], _array = [], _filter = [];
 if (_text == "") {
 changeDataSource(localData);
 }
 else {
 let predicate = new Predicate('name', 'contains', _text, true);
 let filteredList = new DataManager(localData).executeLocal(new
Query().where(predicate));
 for (let j = 0; j < filteredList.length; j++) {
 _filter.push(filteredList[j]["id"]);
 let filters = getFilterItems(filteredList[j], localData);
 for (let i = 0; i < filters.length; i++) {
 if (_array.indexOf(filters[i]) == -1 && filters[i] !=
null) {
 _array.push(filters[i]);
 predicats.push(new Predicate('id', 'equal',
filters[i], false));
 }
 }
 if (predicats.length == 0) {
 changeDataSource([]);
 }
 else {
 let query = new Query().where(new Predicate.or(predicats));
 let newList = new
DataManager(localData).executeLocal(query);
 changeDataSource(newList);
 setTimeout(function () {
 listTreeObj.expandAll();
 }, 400);
 }
 }
 }
}

//Find the Parent Nodes for corresponding childs
function getFilterItems(fList, list) {
 let nodes = [];
 nodes.push(fList["id"]);
 let query2 = new Query().where('id', 'equal', fList["pid"], false);
 let fList1 = new DataManager(list).executeLocal(query2);
 if (fList1.length != 0) {
 let pNode = getFilterItems(fList1[0], list);
 for (let i = 0; i < pNode.length; i++) {
 if (nodes.indexOf(pNode[i]) == -1 && pNode[i] != null)
 nodes.push(pNode[i]);
 }
 return nodes;
 }
}

```

```

 return nodes;
 }

```

## INDEX.HTML

```

<html lang="en"><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head><body>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">

 <div id="container">
 <div style="margin: 20px; padding: 20px; height: 2000px;">
 <div class="treeviewfilter">
 <!-- TextBox Element -->
 <input id="mask" type="text">
 <!-- TreeView Element -->
 <div id="treeView"></div>
 </div>
 </div>
</div>
<style>
 .treeviewfilter {
 width: 200px;
 min-height: 300px;
 max-height: auto;
 border: 1px solid #bbbbbb;
 margin: 0 auto;
 }
 .e-mask .e-input {
 border: none;
 border-bottom: 1px solid #bbbbbb;
 }
 .heading {
 font-size: 22px;
 margin: 0 auto;
 width: 300px;
 margin-bottom: 10px;
 }
</style>

```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### Set tool tip for tree nodes in EJ2 JavaScript Treeview control

TreeView control allows you to set tooltip option to tree nodes using the [tooltip](#) property. The following code example demonstrates how to set tooltip for TreeView nodes.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * TreeView tooltip sample
 */
// Hierarchical data source for TreeView component
let hierarchicalData: { [key: string]: Object }[] = [
 { id: '01', name: 'Local Disk (C:)', expanded: true, tooltip: 'This is
parent directory',
 subChild: [
 {
 id: '01-01', name: 'Program Files', tooltip: 'This is child
directory',
 subChild: [
 { id: '01-01-01', name: 'Windows NT' },
 { id: '01-01-02', name: 'Windows Mail' },
 { id: '01-01-03', name: 'Windows Photo Viewer' },
]
 },
 {
 id: '01-02', name: 'Users', expanded: true, tooltip: 'This
is child directory',
 subChild: [
 { id: '01-02-01', name: 'Smith' },
 { id: '01-02-02', name: 'Public' },
 { id: '01-02-03', name: 'Admin' },
]
 },
 {
 id: '01-03', name: 'Windows', tooltip: 'This is child
directory',
 subChild: [
 { id: '01-03-01', name: 'Boot' },
 { id: '01-03-02', name: 'FileManager' },
 { id: '01-03-03', name: 'System32' },
]
 }
]
 },
 {
```

```

 id: '02', name: 'Local Disk (D:)', tooltip: 'This is parent
directory',
 subChild: [
 {
 id: '02-01', name: 'Personals', tooltip: 'This is child
directory',
 subChild: [
 { id: '02-01-01', name: 'My photo.png' },
 { id: '02-01-02', name: 'Rental document.docx' },
 { id: '02-01-03', name: 'Pay slip.pdf' },
]
 },
 {
 id: '02-02', name: 'Projects', tooltip: 'This is child
directory',
 subChild: [
 { id: '02-02-01', name: 'ASP Application' },
 { id: '02-02-02', name: 'TypeScript Application' },
 { id: '02-02-03', name: 'React Application' },
]
 },
 {
 id: '02-03', name: 'Office', tooltip: 'This is child
directory',
 subChild: [
 { id: '02-03-01', name: 'Work details.docx' },
 { id: '02-03-02', name: 'Weekly report.docx' },
 { id: '02-03-03', name: 'Wish list.csv' },
]
 },
],
 },
 {
 id: '03', name: 'Local Disk (E:)', icon: 'folder', tooltip: 'This is
parent directory',
 subChild: [
 {
 id: '03-01', name: 'Pictures', tooltip: 'This is child
directory',
 subChild: [
 { id: '03-01-01', name: 'Wind.jpg' },
 { id: '03-01-02', name: 'Stone.jpg' },
 { id: '03-01-03', name: 'Home.jpg' },
]
 },
 {
 id: '03-02', name: 'Documents', tooltip: 'This is child
directory',
 subChild: [
 { id: '03-02-01', name: 'Environment Pollution.docx' },
 { id: '03-02-02', name: 'Global Warming.ppt' },
 { id: '03-02-03', name: 'Social Network.pdf' },
]
 },
 {
 id: '03-03', name: 'Study Materials', tooltip: 'This is
child directory',
 }
]
 }
]

```

```

 subChild: [
 { id: '03-03-01', name: 'UI-Guide.pdf' },
 { id: '03-03-02', name: 'Tutorials.zip' },
 { id: '03-03-03', name: 'TypeScript.7z' },
],
 },
]
}
];
let tree1: TreeView = new TreeView({
 fields: { dataSource: hierarchicalData, id: 'id', text: 'name', child:
'subChild' }
});
tree1.appendTo('#tree');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
</script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Sorting treeview level wise in EJ2 JavaScript Treeview control

You can sort the TreeView nodes based on their level. When using the `sortOrder` property, the whole TreeView is sorted. When you sort a particular level, you can use the following code sample. The following code sample demonstrates how to sort the parent node alone in TreeView.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
import { DataManager, Query } from '@syncfusion/ej2-data';
enableRipple(true);

/**
 * TreeView Sort treeview level wise
 */
// List data source for TreeView component
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'India', hasChild: true },
 { id: 2, pid: 1, name: 'Assam' },
 { id: 3, pid: 1, name: 'Bihar' },
 { id: 4, pid: 1, name: 'Tamil Nadu' },
 { id: 6, pid: 1, name: 'Punjab' },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'France', hasChild: true },
 { id: 12, pid: 11, name: 'Pays de la Loire' },
 { id: 13, pid: 11, name: 'Aquitaine' },
 { id: 14, pid: 11, name: 'Brittany' },
 { id: 15, pid: 11, name: 'Lorraine' },
 { id: 16, name: 'Australia', hasChild: true },
 { id: 17, pid: 16, name: 'New South Wales' },
 { id: 18, pid: 16, name: 'Victoria' },
 { id: 19, pid: 16, name: 'South Australia' },
 { id: 20, pid: 16, name: 'Western Australia' },
 { id: 21, name: 'China', hasChild: true },
 { id: 22, pid: 21, name: 'Guangzhou' },
 { id: 23, pid: 21, name: 'Shanghai' },
 { id: 24, pid: 21, name: 'Beijing' },
 { id: 25, pid: 21, name: 'Shantou' }
];

let tree1: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', parentID: 'pid', text:
 'name', hasChildren: 'hasChild' },
 created: onCreate,
 nodeExpanding: onNodeExpand
});

tree1.appendTo('#tree');
function changeDataSource(data) {
 tree1.fields = {
 dataSource: data, id: 'id', text: 'name', parentID: 'pid',
 hasChildren: 'hasChild'
 }
}
let newData;
function onCreate() {
```

```

newData = this.fields.dataSource;
// Selects the first level nodes alone
let resultData = new DataManager(this.getTreeData()).executeLocal(new
Query().where(this.fields.parentID, 'equal', undefined, false));
let name = [];
for (let i = 0; i < resultData.length; i++){
 name.push(resultData[i][this.fields.text]);
}
name.sort();
let arr = [];
for (let j = 0; j < name.length; j++) {
 let sortedData = new
DataManager(this.getTreeData()).executeLocal(new
Query().where(this.fields.text, 'equal', name[j], false));
 let childData = new DataManager(newData).executeLocal(new
Query().where(this.fields.parentID, 'equal',
parseInt(sortedData[0][this.fields.id]), false));
 arr.push(sortedData[0]);
}
// Renders treeview with sorted Nodes
changeDataSource(arr);
tree1.dataBind();
}
function onNodeExpand(args: NodeExpandEventArgs){
 if (args.isInteracted && args.node.querySelector('li') === null){
 let childData = new DataManager(newData).executeLocal(new
Query().where(this.fields.parentID, 'equal', parseInt(args.nodeData.id),
false));
 tree1.addNodes(childData, args.node, null)
 }
}
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

```

```

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### Remove parent checkbox in EJ2 JavaScript Treeview control

By enabling the `showCheckBox` property, you can render check box before each node of TreeView. However, some application needs to render check box in child nodes alone. In such case, you can remove the check box of the parent node by customizing the CSS.

#### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * Removing checkbox of parent nodes TreeView sample
 */
// List data source for TreeView component
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'India', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'Assam' },
 { id: 3, pid: 1, name: 'Bihar' },
 { id: 4, pid: 1, name: 'Tamil Nadu' },
 { id: 6, pid: 1, name: 'Punjab' },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'France', hasChild: true },
 { id: 12, pid: 11, name: 'Pays de la Loire' },
 { id: 13, pid: 11, name: 'Aquitaine' },
 { id: 14, pid: 11, name: 'Brittany' },
 { id: 15, pid: 11, name: 'Lorraine' },
 { id: 16, name: 'Australia', hasChild: true },
 { id: 17, pid: 16, name: 'New South Wales' },
 { id: 18, pid: 16, name: 'Victoria' },
 { id: 19, pid: 16, name: 'South Australia' },
 { id: 20, pid: 16, name: 'Western Australia' },
 { id: 21, name: 'China', hasChild: true },
 { id: 22, pid: 21, name: 'Guangzhou' },
 { id: 23, pid: 21, name: 'Shanghai' },
 { id: 24, pid: 21, name: 'Beijing' },
 { id: 25, pid: 21, name: 'Shantou' }
];
let tree1: TreeView = new TreeView({

```



```

 fields: { dataSource: countries, id: 'id', text: 'name', parentID:
'pid', hasChildren: 'hasChild' },
 showCheckBox: true,
 cssClass: 'custom'
 });
 tree1.appendTo('#tree');

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## INDEX.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 width: 30%;
 position: absolute;

```

```

 font-family: 'Helvetica Neue','calibiri';
 font-size: 14px;
 top: 45%;
 left: 45%;
}
#treeparent {
 display: block;
 max-width: 350px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
}
.custom .e-list-item.e-level-1 .e-text-content.e-icon-wrapper
 .e-checkbox-wrapper {
 display: none
 }

```

### Get dynamic icon in EJ2 JavaScript Treeview control

In TreeView component, you can get the original bound data using the `getTreeData` method. For this method, if you pass the id of the tree node, it returns the corresponding node information, or otherwise the overall tree nodes information will be returned. You can use this method to get the bound iconCss class in the `nodeChecking` event. Please refer to the following sample.

#### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * TreeView dynamic iconCss sample
 */
// List data source for TreeView component
let treeData: { [key: string]: Object }[] = [
 {
 "nodeId": "01", "nodeText": "Music", "icon": "folder", "expanded": true,
 "nodeChild": [
 { "nodeId": "01-01", "nodeText": "Gouttes.mp3", "icon": "audio" }
]
 },
 {
 "nodeId": "02", "nodeText": "Videos", "icon": "folder", "expanded":
true,
 "nodeChild": [
 { "nodeId": "02-01", "nodeText": "Naturals.mp4", "icon": "video" },
 { "nodeId": "02-02", "nodeText": "Wild.mpeg", "icon": "video" }
]
 }
];
let tree1: TreeView = new TreeView({
 fields: { dataSource: treeData, id: 'nodeId', text: 'nodeText', child:
'nodeChild', iconCss: 'icon', expanded: 'expanded' },
 showCheckBox: true,
 nodeChecking: onNodeChecking,

```

```
 autoCheck: false
 });
 tree1.appendTo('#tree');
 function onNodeChecking(args) {
 let nodeId = args.data[0].id;
 // To get the iconCss
 let iconClass = this.getTreeData(nodeId)[0].icon;
 alert('Icon class is ' + iconClass);
 }
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## INDEX.CSS

```
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
```

```

height: 40px;
width: 30%;
position: absolute;
font-family: 'Helvetica Neue','calibiri';
font-size: 14px;
top: 45%;
left: 45%;
}
#treeparent {
 display: block;
 max-width: 350px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
}
.e-treeview .e-list-img {
 width: 25px;
 height: 25px;
}
.e-treeview .e-list-icon {
 background-repeat: no-repeat;
 background-image:
url(https://ej2.syncfusion.com/javascript/demos/src/treeview/images/icons/fi
le_icons.png);
 height: 20px;
}
.e-treeview .e-list-icon.folder { background-position: -10px -552px }
.e-treeview .e-list-icon.audio { background-position: -10px -244px }
.e-treeview .e-list-icon.video { background-position: -10px -272px }

```

### Hover multi line tree node in EJ2 JavaScript Treeview control

This section demonstrates how to hover and select a multi-line tree node. Here, you can set the row height (element class: `e-fullrow`) to be the same as the row content (element class: `e-text-content`)

#### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * Hovering multiple line TreeView sample
 */
// List data source for TreeView component
let hierarchicalData: { [key: string]: Object }[] = [
 {
 id: 1, name: 'Web Control sWeb ControlsWeb ControlsWeb
ControlsWeb ControlsWeb ControlsWeb ControlsWeb Controls', expanded: true,
 child: [
 {
 id: 2, pid: 1, name:
'CalendarCalendarCalendarCalendarCalendarCalendarCalendarCalendarCal
endarCalendarCalendarCalendar', checked: true, child: [
 { id: 7, pid: 2, name: 'Constructors' },
 { id: 8, pid: 2, name: 'Properties' },

```

```

 { id: 9, pid: 2, name: 'Methods' },
 { id: 10, pid: 2, name: 'Events' }
],
 },
 {
 id: 3, pid: 1, name: 'Data Grid', child: [
 { id: 11, pid: 3, name: 'Constructors' },
 { id: 12, pid: 3, name: 'Fields' },
 { id: 13, pid: 3, name: 'Properties' },
 { id: 14, pid: 3, name: 'Methods' },
 { id: 15, pid: 3, name: 'Events' }
]
 },
 {
 id: 4, pid: 1, name: 'DropDownList', child: [
 { id: 16, pid: 4, name: 'Constructors' },
 { id: 17, pid: 4, name: 'Properties' },
 { id: 18, pid: 4, name: 'Methods' }
]
 },
 {
 id: 5, pid: 1, name: 'Menu', child: [
 { id: 19, pid: 5, name: 'Constructors' },
 { id: 20, pid: 5, name: 'Fields' },
 { id: 21, pid: 5, name: 'Properties' },
 { id: 22, pid: 5, name: 'Methods' },
 { id: 23, pid: 5, name: 'Events' }
]
 }
],
},
{
 id: 24, name: 'Web Controls',
 child: [
 {
 id: 25, pid: 24, name: 'Calendar', checked: true, child:
[
 { id: 26, pid: 25, name: 'Constructors' },
 { id: 27, pid: 25, name: 'Properties' },
 { id: 28, pid: 25, name: 'Methods' },
 { id: 29, pid: 25, name: 'Events' }
]
 },
 {
 id: 30, pid: 24, name: 'Data Grid', child: [
 { id: 31, pid: 30, name: 'Constructors' },
 { id: 32, pid: 30, name: 'Fields' },
 { id: 33, pid: 30, name: 'Properties' },
 { id: 34, pid: 30, name: 'Methods' },
 { id: 35, pid: 30, name: 'Events' }
]
 }
]
}
];
// Render the TreeView by mapping its fields property with data source
properties

```

```

let treeObj: TreeView = new TreeView({
 fields: { dataSource:hierarchicalData , id: 'id', text: 'name',
child: 'child' },
 nodeSelecting: onselect,
 cssClass: ("customTree")
});
treeObj.appendTo('#tree');
// Triggers on mouse hover/keydown event
['mouseover','keydown'].forEach(evt =>
 document.getElementById("tree").addEventListener(evt,
(event)=>{setHeight(event.target); }));
// Triggers on node selection
function onSelect(arg) {
 setHeight(arg.node);
}
// Sets e-fullrow to be the same as e-text-content
function setHeight(element) {
 if(treeObj.fullRowSelect) {
 if(element.classList.contains("e-treeview")) {
 element = element.querySelector(".e-node-
focus").querySelector(".e-fullrow");
 }
 else if(element.classList.contains("e-list-parent")) {
 element = element.querySelector(".e-fullrow");
 }
 else if(element.classList.value != ("e-fullrow") &&
element.closest(".e-list-item")) {
 element = element.closest(".e-list-item").querySelector(".e-
fullrow");
 }
 if(element.nextElementSibling)
 element.style.height = element.nextElementSibling.offsetHeight
+"px";
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>

```

```

<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### INDEX.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 width: 30%;
 position: absolute;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 top: 45%;
 left: 45%;
}
#treeparent {
 display: block;
 max-width: 350px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
}
.customTree li .e-list-text{
 white-space: normal;
 word-break: break-all;
}

```

### Select one child in EJ2 JavaScript Treeview control

TreeView allows both single and multiple selections. If your application needs to select one child at a time under one specific parent, refer to the following example. Here, you can achieve this in the `nodeSelecting` event of TreeView. However, you can reset the selected child and make another selection by pressing Ctrl + selected nodes.

### INDEX.TS

```

import { TreeView } from '@syncfusion/ej2-navigations';
/**
 * Single child selection at a time
 */
// List data source for TreeView component
let localData: { [key: string]: Object }[] = [
 { id: 1, name: 'Parent 1', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'Child 1' },
 { id: 3, pid: 1, name: 'Child 2' },
 { id: 4, pid: 1, name: 'Child 3' },
 { id: 7, name: 'Parent 2', hasChild: true, expanded: true },
 { id: 8, pid: 7, name: 'Child 1' },
 { id: 9, pid: 7, name: 'Child 2' },
 { id: 10, pid: 7, name: 'Child 3' },
];
let tree1: TreeView = new TreeView({
 fields: { dataSource: localData, id: 'id', text: 'name', parentID:
'pid', hasChildren: 'hasChild' },
 loadOnDemand: false,
 allowMultiSelection: true
 nodeSelecting: onNodeSelecting
});
tree1.appendTo('#tree');
let parent, child;
let count: boolean = false;
let childCount: boolean = false;
function onNodeSelecting(args: NodeSelectEventArgs) {
 let id = args.nodeData.parentID;
 if (!count) {
 parent = id;
 count = true;
 }
 if (!childCount) {
 child = args.nodeData.id;
 childCount = true
 }
 if (id !== null && id === parent) {
 let element: HTMLElement = tree1.element.querySelector('[data-uid="' +
id + '"]');
 let liElements = element.querySelectorAll('ul li');
 for (let i: number = 0; i < liElements.length; i++) {
 let nodeData = tree1.getNode(liElements[i]);
 if (nodeData.selected && args.action === "select" && child !==
args.nodeData.id) {
 args.cancel = true;
 }
 // For unselect the selectedNodes
 else if (args.action === "un-select" && child === args.nodeData.id)
{
 childCount = false;
 child = null;
 parent = null;
 count = false;
 }
 }
 }
 else if (id !== parent && id !== null) {
 if(args.action === "select"){

```



```

 args.cancel = true
 }
 } else if (id === null) {
 childCount = false;
 child = null;
 parent = null;
 count = false
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

### Get all child nodes in EJ2 JavaScript Treeview control

This section demonstrates how to get the child nodes from corresponding parent ID. Using the `getNode` method, you can get the node details of TreeView. Please refer to the following sample.

## INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * TreeView get child nodes from parent id sample
 */
// List data source for TreeView component
let data: { [key: string]: Object }[] = [
{
 code: "AF", name: "Africa", countries: [
 { code: "NGA", name: "Nigeria" },
 { code: "EGY", name: "Egypt" },
 { code: "ZAF", name: "South Africa" }
]
},
{
 code: "AS", name: "Asia", countries: [
 { code: "CHN", name: "China" },
 { code: "IND", name: "India", countries: [
 {code: "TN", name: "TamilNadu" }
]},
 { code: "JPN", name: "Japan" }
]
},
{
 code: "EU", name: "Europe", countries: [
 { code: "DNK", name: "Denmark" },
 { code: "FIN", name: "Finland" },
 { code: "AUT", name: "Austria" }
]
},
{
 code: "NA", name: "North America", countries: [
 { code: "USA", name: "United States of America" },
 { code: "CUB", name: "Cuba" },
 { code: "MEX", name: "Mexico" }
]
},
{
 code: "SA", name: "South America", countries: [
 { code: "BR", name: "Brazil" },
 { code: "COL", name: "Colombia" },
 { code: "ARG", name: "Argentina" }
]
},
];
let tree1: TreeView = new TreeView({
 fields: { dataSource: data, id: 'code', text: 'name', child: 'countries'
},
 loadOnDemand: false,
 created: onCreate
});
tree1.appendTo('#tree');
function onCreate() {
 let proxy = this;
 document.getElementById("btn").addEventListener("click", (event)=>{
 let id = document.getElementById('Nodes').value

```

```

 let element= proxy.element.querySelector('[data-uid="' + id +
 '"]');
 // Gets the child Element
 let liElements = element.querySelectorAll('ul li');
 let arr= [];
 for (let i = 0; i < liElements.length; i++) {
 let nodeData= proxy.getNode(liElements[i]);
 arr.push(nodeData);
 }
 alert(JSON.stringify(arr));
 })
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>

 <input type="text" class="e-input" id="Nodes" style="margin-left:
300px; width: 175px;" placeholder="Enter the parent ID(Ex: AS)">
 <input type="button" class="btn btn-primary" value="Submit"
id="btn">

 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
<script src="index.js" type="text/javascript"></script>

```

```
</body></html>
```

### Disable checkbox of the tree node in EJ2 JavaScript Treeview control

You can disable the check box alone in TreeView instead of disabling the whole node. You need to include the `e-checkbox-disabled` class into the check box element using the `drawNode` event. Please refer to the following sample to disable the check box of the tree nodes.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
import { TreeView } from '@syncfusion/ej2-navigations';
enableRipple(true);
/**
 * Treeview Disable check box of parent nodes sample
 */
// List data source for TreeView component
let countries: { [key: string]: Object }[] = [
 { id: 1, name: 'India', hasChild: true, expanded: true },
 { id: 2, pid: 1, name: 'Assam' },
 { id: 3, pid: 1, name: 'Bihar' },
 { id: 4, pid: 1, name: 'Tamil Nadu' },
 { id: 6, pid: 1, name: 'Punjab' },
 { id: 7, name: 'Brazil', hasChild: true },
 { id: 8, pid: 7, name: 'Paraná' },
 { id: 9, pid: 7, name: 'Ceará' },
 { id: 10, pid: 7, name: 'Acre' },
 { id: 11, name: 'France', hasChild: true },
 { id: 12, pid: 11, name: 'Pays de la Loire' },
 { id: 13, pid: 11, name: 'Aquitaine' },
 { id: 14, pid: 11, name: 'Brittany' },
 { id: 15, pid: 11, name: 'Lorraine' },
 { id: 16, name: 'Australia', hasChild: true },
 { id: 17, pid: 16, name: 'New South Wales' },
 { id: 18, pid: 16, name: 'Victoria' },
 { id: 19, pid: 16, name: 'South Australia' },
 { id: 20, pid: 16, name: 'Western Australia' },
 { id: 21, name: 'China', hasChild: true },
 { id: 22, pid: 21, name: 'Guangzhou' },
 { id: 23, pid: 21, name: 'Shanghai' },
 { id: 24, pid: 21, name: 'Beijing' },
 { id: 25, pid: 21, name: 'Shantou' }
];
// Renders treeview
let tree1: TreeView = new TreeView({
 fields: { dataSource: countries, id: 'id', text: 'name', parentID:
'pid', hasChildren: 'hasChild' },
 showCheckBox: true,
 drawNode: drawNode
});
tree1.appendTo('#tree');
// Disables the checkbox alone in treeview
function drawNode(args: DrawNodeEventArgs) {
 let ele: HTMLElement = args.node.querySelector('.e-checkbox-wrapper');
 ele.classList.add('e-checkbox-disabled');
}
```

**INDEX.HTML**

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 for TreeView </title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 for TreeView UI
Control">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="treeparent">
 <div id="tree"></div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

[Ej1 api migration in EJ2 JavaScript Treeview control](#)

This article describes the API migration process of TreeView component from Essential JS 1 to Essential JS 2.

[Add nodes](#)

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Add node | **Method:** `addNode`  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});`  
`var treeObj = $("#tree").data("ejTreeView");`  
`treeObj.addNode("Node", "#book");` | **Method:** `addNodes`  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren:`

```
"hasChild"},
});
treeObj.appendTo("#tree");
var object = [{ id: "temp", name: "New node"
}, { id: "new", name: "New node 1" }];
treeObj.addNodes(object, "book"); |
```

| Triggers before adding node | **Event:** *beforeAdd*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>beforeAdd: function() {}<br/>}); | Not Applicable |

| Adding node after a particular node | **Method:**

```
insertAfter

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid",
text: "name", hasChild: "hasChild"}
});
var treeObj =
$("#tree").data("ejTreeView");
treeObj.insertAfter("node", "book"); | Can be achieved
using

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChildren:
"hasChild"},
});
treeObj.appendTo("#tree");
var object = [{ id: "1", name: "node"
}];
var child = treeObj.element.querySelector('[data-uid="book"]');
var parent =
child.parentElement.closest('.e-list-item');
var level = parseInt(parent.getAttribute('aria-
level'))+1;
var childNodes = Array.from(parent.querySelectorAll('.e-list-item.e-level-
'+level))
var index = childNodes.indexOf(child)
treeObj.addNodes(object, "book", index+1); |
```

| Adding node before a particular node | **Method:**

```
insertBefore

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId:
"pid", text: "name", hasChild: "hasChild"}
});
var treeObj =
$("#tree").data("ejTreeView");
treeObj.insertBefore("node", "book"); | Can be achieved
using

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChildren:
"hasChild"},
});
treeObj.appendTo("#tree");
var object = [{ id: "1", name: "node"
}];
var child = treeObj.element.querySelector('[data-uid="book"]');
var parent =
child.parentElement.closest('.e-list-item');
var level = parseInt(parent.getAttribute('aria-
level'))+1;
var childNodes = Array.from(parent.querySelectorAll('.e-list-item.e-level-
'+level))
var index = childNodes.indexOf(child)
treeObj.addNodes(object, "book", index-1); |
```

| Triggers when node is added successfully | **Event:**

```
nodeAdd

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid",
text: "name", hasChild: "hasChild"},
nodeAdd: function() {}
}); | Event:
dataSourceChanged

var treeObj = new ej.navigations.TreeView({
fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},
dataSourceChanged:
function() {}
});
treeObj.appendTo("#tree"); |
```

## Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Keyboard Navigation | **Property:**

```
allowKeyboardNavigation

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChild: "hasChild"},
allowKeyboardNavigation: false
}); | Can
be achieved using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},
keyPress: function(args)
{
args.cancel = true;
}
});
treeObj.appendTo("#tree"); |
```

| Triggers before node is cut | **Event:** *beforeCut*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
function() {}  
| Not Applicable |

| Triggers before node is deleted | **Event:** *beforeDelete*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
function() {}  
| Not Applicable |

| Triggers before loading nodes | **Event:** *beforeLoad*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
function() {}  
| Not Applicable |

| Triggers before node is pasted | **Event:** *beforePaste*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
function() {}  
| Not Applicable |

| Triggers when Treeview is created | **Event:** *create*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
function() {}  
| **Event:** *created*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
function() {}  
treeObj.appendTo("#tree"); |

| Css class | **Property:** *cssClass*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
cssClass: 'custom'  
| **Property:** *cssClass*  
var treeObj = new ej.navigations.TreeView(  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
cssClass: 'custom'  
);  
treeObj.appendTo("#tree"); |

| Triggers when Treeview is destroyed | **Event:** *destroy*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
function() {}  
| **Event:** *destroyed*  
var treeObj = new  
ej.navigations.TreeView(  
{dataSource: treeData, id: "id", parentId: "pid", text: "name",  
hasChildren: "hasChild"},  
destroyed: function() {}  
);  
treeObj.appendTo("#tree"); |

| Destroy Treeview control | **Method:** *destroy*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}  
var treeObj = \$("#tree").data("ejTreeView");  
treeObj.destroy(); | **Method:** *destroy*  
var treeObj = new ej.navigations.TreeView(  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}  
);  
treeObj.appendTo("#tree");  
treeObj.destroy(); |

| Disable Node | **Method:** *disableNode*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}  
var treeObj = \$("#tree").data("ejTreeView");  
treeObj.disableNode(\$("#1")); | **Method:** *disableNodes*  
var treeObj = new ej.navigations.TreeView(  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}  
);  
treeObj.appendTo("#tree");  
treeObj.disableNodes(["1", "2"]); |

| Enable Animation | **Property:** *enableAnimation*  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
enableAnimation: false  
| **Property:** *animation*  
var treeObj = new  
ej.navigations.TreeView(  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
animation: {expand: {duration: 0}, collapse: {duration: 0}}  
);  
treeObj.appendTo("#tree"); |

| Control state | **Property:** *enabled*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, enabled: false}); | Not Applicable |

| Enable Node | **Method:** *enableNode*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.enableNode( "#1" ); | **Method:** *enableNodes*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 treeObj.enableNodes( [ "1", "2" ] ); |

| Persistence | **Property:** *enablePersistence*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, enablePersistence: true}); | **Property:** *enablePersistence*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, enablePersistence: true});  
 treeObj.appendTo( "#tree" ); |

| Right to Left | **Property:** *enableRTL*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, enableRTL: true}); | **Property:** *enableRtl*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, enableRtl: true});  
 treeObj.appendTo( "#tree" ); |

| Ensure visibility | **Method:** *ensureVisible*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.ensureVisible( "#1" ); | **Method:** *ensureVisible*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 treeObj.ensureVisible( "1" ); |

| Mapping fields | **Property:** *fields*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: [], id: "id", parentId: "pid", text: "name", child: "child", hasChild: "hasChild", expanded: "expanded", htmlAttribute: "htmlAttributes", imageAttribute: "img", imageUrl: "imageUrl", isChecked: "checked", linkAttribute: "linkAttribute", query: null, selected: "selected", spriteCssClass: "custom", tableName: null}}); | **Property:** *fields*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: [], id: "id", text: "name", child: 'child', parentId: 'pid', hasChildren: 'hasChild', expanded: 'expanded', htmlAttributes: 'htmlAttributes', iconCss: 'iconCss', imageUrl: 'imageUrl', isChecked: 'checked', query: null, selected: 'selected', tableName: null, tooltip: 'tooltip', navigateUrl: 'navigateUrl'}});  
 treeObj.appendTo( "#tree" ); |

| Get child nodes | **Method:** *getChildren*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.getChildren( "1" ); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 var parent=treeObj.element.querySelector( '[data-uid="1"]' );  
 console.log( parent.querySelector( '.e-list-item' ) ); |

| Get node | **Method:** *getNode*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.getNode( "#1" ); | **Method:** *getNode*  
 var



```
treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}
});
treeObj.appendTo("#tree");
treeObj.getNode("1"); |
```

```
| Get node by index | Method: getNodeByIndex

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.getNodeByIndex(3); | Can be achieved using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}
});
treeObj.appendTo("#tree");
var nodes=treeObj.element.querySelectorAll('.e-list-item')
console.log(nodes[3]); |
```

```
| Get node count | Method: getNodeCount

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.getNodeCount(); | Can be achieved using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}
});
treeObj.appendTo("#tree");
var nodes=treeObj.element.querySelectorAll('.e-list-item')
console.log("Node count is " + nodes.length); |
```

```
| Get node index | Method: getNodeIndex

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.getNodeIndex($("#book")); | Can be achieved using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}
});
treeObj.appendTo("#tree");
var nodes=treeObj.element.querySelectorAll('.e-list-item')
var node = treeObj.element.querySelector('[data-uid="" + "book" + "']');
while(i<nodes.length) {
if(nodes[i] === node) {console.log("Node index is " + i)
break;
}
i++
} |
```

```
| Get parent of node | Method: getParent

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.getParent($("#book")); | Can be achieved using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}
});
treeObj.appendTo("#tree");
var child=treeObj.element.querySelector('[data-uid="" + "book" + "']');
var parent = child.parentNode.closest('.e-list-item')
console.log(parent) |
```

```
| Get tree node text | Method: getText

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.getText("1"); | Can be achieved using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}
});
treeObj.appendTo("#tree");
var nodeText =treeObj.getNode("1")['text']
console.log(nodeText) |
```

```
| Get updated datasource | Method: getTreeData

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.getTreeData(); | Method: getTreeData

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}
});
treeObj.appendTo("#tree");
treeObj.getTreeData(); |
```

| Get visible nodes | **Method:** *getVisibleNodes*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}  
 treeObj = \$("#tree").data("ejTreeView"); treeObj.getVisibleNodes(); | Not Applicable |

| Height of Treeview control | **Property:** *height*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 "400px"; | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name",  
 hasChildren: "hasChild"},  
 "custom"); treeObj.appendTo("#tree");  
 \$.e-treeview.custom{height: 400px};

| Checking for child nodes | **Method:** *hasChildNode*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}  
 treeObj = \$("#tree").data("ejTreeView"); treeObj.hasChildNode(\$("#book")); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id",  
 parentId: "pid", text: "name", hasChildren: "hasChild"}); treeObj.appendTo("#tree");  
 parent=treeObj.element.querySelector('[data-uid="book"]'); if (parent.querySelector('.e-list-item')  
 !== null) {console.log("Has child node")}

| Hide all nodes | **Method:** *hide*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}  
 treeObj = \$("#tree").data("ejTreeView"); treeObj.hide(); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name",  
 hasChildren: "hasChild"}); treeObj.appendTo("#tree"); treeObj.element.querySelector('.e-list-parent').style.display="none"

| Hide node | **Method:** *hideNode*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}  
 treeObj = \$("#tree").data("ejTreeView"); treeObj.hideNode(\$("#book")); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id",  
 parentId: "pid", text: "name", hasChildren: "hasChild"}); treeObj.appendTo("#tree"); treeObj.element.querySelector('[data-uid="book"]').style.display="none"

| HTML Attributes | **Property:** *htmlAttributes*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 htmlAttributes: {class: "htmlAttr"}; | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name",  
 hasChildren: "hasChild"}); treeObj.appendTo("#tree"); treeObj.element.classList.add("htmlAttr")

| To check if child nodes are loaded | **Method:** *isChildLoaded*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}  
 treeObj = \$("#tree").data("ejTreeView"); treeObj.isChildLoaded(\$("#book")); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id",  
 parentId: "pid", text: "name", hasChildren: "hasChild"}); treeObj.appendTo("#tree");  
 parent=treeObj.element.querySelector('[data-uid="book"]'); if (parent.querySelector('.e-list-item')  
 !== null) {console.log("Child is loaded")}

| To check if node is disabled | **Method:** *isDisabled*  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
var treeObj = $("#tree").data("ejTreeView");  
treeObj.isDisabled($("#book"));` | **Can be achieved using,**  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
treeObj.appendTo("#tree");  
var node=treeObj.element.querySelector('[data-uid="book"]');  
if (node.classList.contains('e-disable') === true) {  
console.log("Node is disabled");  
}` |

| To check if node exists in Treeview | **Method:** *isExist*  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
var treeObj = $("#tree").data("ejTreeView");  
treeObj.isExist($("#book"));` | **Can be achieved using,**  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
treeObj.appendTo("#tree");  
if (treeObj.getNode('book')['text'] !== "") {  
console.log("Node exists");  
}` |

| To check if node is visible | **Method:** *isVisible*  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
var treeObj = $("#tree").data("ejTreeView");  
treeObj.isVisible($("#book"));` | **Can be achieved using,**  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
treeObj.appendTo("#tree");  
if (treeObj.element.querySelector('[data-uid="book"]').style.display !== "none"){  
console.log("Node is visible");  
}` |

| Triggers on key press | **Event:** *keyPress*  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
keyPress: function() {}  
});` | **Event:** *keyPress*  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
keyPress: function() {}  
});  
treeObj.appendTo("#tree");` |

| Load Treeview nodes from particular URL | **Method:** *loadData*  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
var treeObj = $("#tree").data("ejTreeView");  
treeObj.loadData("childData", $("#book"));` | **Can be achieved using,**  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
treeObj.appendTo("#tree");  
var dataManager = new DataManager {Url = "/FileContent/rootNode",  
Adaptor = "UrlAdaptor",  
CrossDomain = true};  
dataManager.executeQuery(new ej.data.Query().take(8).then((e) => { var childData = e.result;  
treeObj.addNodes(childData, "book");  
}));` |

| Triggers when data load fails | **Event:** *loadError*  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
loadError: function() {}  
});` | **Can be achieved using,**  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
treeObj.appendTo("#tree");  
var dataManager = new DataManager {Url = "/FileContent/rootNode",  
Adaptor = "UrlAdaptor",  
CrossDomain = true};  
dataManager.executeQuery(new ej.data.Query().take(8).error((e) => { console.log('Data load failed');  
}));` |

| Load on demand | **Property:** *loadOnDemand*  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild:`

"hasChild"},<br/>loadOnDemand: true<br/>}); | **Property:** *loadOnDemand*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>loadOnDemand: false<br/>});<br/>treeObj.appendTo("#tree");<br/>**Treeview is rendered in load on demand by default**

| Triggers when data load is success | **Event:** *loadSuccess*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>loadSuccess: function() {}<br/>});<br/>**Script**<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.loadData("childData", \$("#book")); | **Can be achieved using**,<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}<br/>});<br/>treeObj.appendTo("#tree");<br/>var dataManager = new DataManager {<br />Url = "/FileContent/rootNode",<br />Adaptor = "UrlAdaptor",<br />CrossDomain = true};<br />dataManager.executeQuery(new ej.data.Query().take(8).then((e) => { console.log('Data loaded successfully')<br/>})); |

| To move node | **Method:** *moveNode*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.moveNode(\$("#book"), "#art"); | **Method:** *moveNodes*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>});<br/>treeObj.appendTo("#tree");<br/>treeObj.moveNodes(["book"], "#art"); |

| Triggers when node is clicked successfully | **Event:** *nodeClick*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>nodeClick: function() {}<br/>}); | **Event:** *nodeClicked*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>nodeClicked: function() {}<br/>});<br/>treeObj.appendTo("#tree"); |

| Triggers when node is cut successfully | **Event:** *nodeCut*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>nodeCut: function() {}<br/>}); | Not Applicable |

| Triggers when node is deleted successfully | **Event:** *nodeDelete*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>nodeDelete: function() {}<br/>}); | **Event:** *dataSourceChanged*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>dataSourceChanged: function() {}<br/>});<br/>treeObj.appendTo("#tree"); |

| Triggers when node is pasted successfully | **Event:** *nodePaste*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>nodePaste: function() {}<br/>}); | Not Applicable |

| Triggers when nodes are loaded successfully | **Event:** *ready*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>ready: function() {}<br/>}); | **Event:** *dataBound*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>dataBound: function(args) {}<br/>});<br/>treeObj.appendTo("#tree"); |

| Refresh Treeview control | **Method:** *refresh*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.refresh(); | Not Applicable |

| To show all nodes | **Method:** *show*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.show(); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 treeObj.element.querySelector( '.e-list-parent' ).style.display="block" |

| Show node | **Method:** *showNode*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.showNode( \$("#book") ); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 treeObj.element.querySelector( '[data-uid="book"]' ).style.display="block" |

| Remove all Treeview nodes | **Method:** *removeAll*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.removeAll(); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 treeObj.removeNodes( [treeObj.element.querySelector( '.e-list-parent' )] ); |

| Remove Treeview node | **Method:** *removeNode*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.removeNode( \$("#book") ); | **Method:** *removeNodes*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 treeObj.removeNodes( ["book"] ); |

| Sort order | **Property:** *sortSettings*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, sortSettings: {allowSorting: true, sortOrder: ej.sortOrder.Descending}}); | **Property:** *sortOrder*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, sortOrder: "Descending"});  
 treeObj.appendTo( "#tree" ); |

| Update node text | **Method:** *updateText*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}});  
 var treeObj = \$( "#tree" ).data( "ejTreeView" );  
 treeObj.updateText( \$("#book", "text") ); | **Method:** *updateNode*  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}});  
 treeObj.appendTo( "#tree" );  
 treeObj.updateNode( "book", "text" ); |

| Width of Treeview control | **Property:** *width*  
 \$( "#tree" ).ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, width: "300px"}); | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name",

```
hasChildren: "hasChild"},
cssClass:
"custom"
});
treeObj.appendTo("#tree");
Css
.e-treeview.custom{
width:
300px;
} |
```

### CheckBox

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Prevent auto-check of child and parent | **Property:**

```
autoCheck

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid",
text: "name", hasChild: "hasChild"},
showCheckbox: true,
autoCheck: false
}); | Property:
autoCheck

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id:
"id", parentId: "pid", text: "name", hasChildren: "hasChild"},
showCheckBox: true,
autoCheck:
false
});
treeObj.appendTo("#tree"); |
```

| Prevent indeterminate state in parent node | **Property:**

```
autoCheckParentNode

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChild: "hasChild"},
showCheckbox:
true,
autoCheckParentNode: true
}); | Can be achieved using,

var treeObj = new
ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name",
hasChildren: "hasChild"},
showCheckBox: true,
autoCheck: false,
nodeChecked:
function(args){
var child = treeObj.element.querySelector('[data-uid="'+ args.data[0]['id'] +
"']);
var checkNodes = [];
var element = child.parentNode;
while ((element !== null ||
element !== undefined) && !element.parentNode.classList.contains('e-treeview')) {
element =
element.parentNode;
var id = element.getAttribute('data-uid');
if (id !== null)
checkNodes.push(element.getAttribute('data-uid'))

if (child.querySelector('.e-list-item') !==
null && args.isInteracted === true && args.action === 'check') {
treeObj.autoCheck =
true;
treeObj.checkAll(child.getAttribute('data-uid'));
} else if (child.querySelector('.e-list-item')
!== null && args.isInteracted === true && args.action === 'uncheck') {
treeObj.autoCheck =
true;
treeObj.uncheckAll(child.getAttribute('data-uid'));

treeObj.autoCheck =
false;
if (args.action === 'check') {
treeObj.checkAll(checkNodes)

else if (args.action
=== 'uncheck' && child.parentNode.querySelector('.e-check') === null)
{
treeObj.uncheckAll(checkNodes)

});
treeObj.appendTo("#tree"); |
```

| Check all nodes | **Method:** *checkAll*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>showCheckbox:
true<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.checkAll(); | **Method:**
*checkAll*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id:
"id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>showCheckBox:
true<br/>});<br/>treeObj.appendTo("#tree");<br/>treeObj.checkAll(); |

| Check node | **Method:** *checkNode*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>showCheckbox:
true<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.checkNode(\$("#book")); | **Method:**
*checkAll*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>showCheckBox:
true<br/>});<br/>treeObj.appendTo("#tree");<br/>treeObj.checkAll(["book"]); |

| Set checkednodes | **Property:** *checkedNodes*<br/><br/>\$("#tree").ejTreeView({<br/>fields:
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>showCheckbox:
true,<br/>checkedNodes: [2, 8, 12]<br/>}); | **Property:** *checkedNodes*<br/><br/>var treeObj = new

```
ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name",
hasChildren: "hasChild"},
showCheckBox: true,
checkedNodes: ["3", "9",
"13"]
});
treeObj.appendTo("#tree"); |
```

| Get checked nodes | **Method:** *getCheckedNodes*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>showCheckbox: true<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.getCheckedNodes(); | **Can be achieved using**,<br/><br/>let treeObj: TreeView = new TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>});<br/>treeObj.appendTo("#tree");<br/>var check=treeObj.getAllCheckedNodes();<br/>var i=0;<br/>var checkedNodes;<br/>while(i<check.length) {<br/>checkedNodes.push(treeObj.element.querySelector("[data-uid=" + checkedNodes[i] + ""]));<br/>i++;<br/>}<br/>console.log(checkedNodes) |

| Get checked nodes index | **Method:** *getCheckedNodesIndex*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>showCheckbox: true<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.getCheckedNodesIndex(); | **Can be achieved using**,<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>showCheckBox: true<br/>});<br/>treeObj.appendTo("#tree");<br/>var nodes=treeObj.element.querySelectorAll('.e-list-item')<br/>var nodeIndex;<br/>while(i<nodes.length) {<br/>if(nodes[i].classList.contains('e-check')) {nodeIndex.push(i);<br/>}<br/>i++;<br/>}<br/>console.log(nodeIndex) |

| To check if nodes are checked | **Method:** *isNodeChecked*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.isNodeChecked(\$("#book")); | **Can be achieved using**,<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>showCheckBox: true<br/>});<br/>treeObj.appendTo("#tree");<br/>if (treeObj.getNode("book")["isChecked"] === true) {<br/>console.log("Node is checked")<br/>} |

| Triggers when node is checked successfully | **Event:** *nodeCheck*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/><br/>showCheckBox: true,<br/>nodeCheck: function() {}<br/>}); | **Event:** *nodeChecked*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/><br/>showCheckBox: true,<br/>nodeChecked: function(args) {<br/>if (args.action == "check") {}<br/>}<br/>});<br/>treeObj.appendTo("#tree"); |

| Checkbox support | **Property:** *showCheckbox*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>showCheckbox: true<br/>}); | **Property:** *showCheckBox*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>showCheckBox: true<br/>});<br/>treeObj.appendTo("#tree"); |

| To uncheck all nodes | **Method:** *unCheckAll*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>showCheckbox: true<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.unCheckAll(); | **Method:** *uncheckAll*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id:

```
"id", parentId: "pid", text: "name", hasChildren: "hasChild"},
showCheckBox:
true
});
treeObj.appendTo("#tree");
treeObj.uncheckAll(); |
```

```
| To uncheck node | Method: uncheckNode

$("#tree").ejTreeView({
fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},
showCheckbox:
true
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.uncheckNode($("#book")); |
Method: uncheckAll

var treeObj = new ej.navigations.TreeView({
fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},
showCheckBox:
true
});
treeObj.appendTo("#tree");
treeObj.uncheckAll(["book"]); |
```

```
| Triggers when node is unchecked successfully | Event:
nodeUncheck

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId:
"pid", text: "name", hasChild: "hasChild"},

showCheckBox: true,
nodeUncheck: function()
{
}); | Event: nodeChecked

var treeObj = new ej.navigations.TreeView({
fields:
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren:
"hasChild"},

showCheckBox: true,
nodeChecked: function(args) {
if (args.action ==
"un-check") {

}}
treeObj.appendTo("#tree"); |
```

```
| Triggers before nodes are checked/ unchecked | Not Applicable | Event: nodeChecking

var
treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text:
"name", hasChildren: "hasChild"},

showCheckBox: true,
nodeChecking: function(args)
{
if (args.action == "check") {

}}
treeObj.appendTo("#tree"); |
```

## Drag and Drop

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```
| Drag and drop | Property: allowDragAndDrop

$("#tree").ejTreeView({
fields:
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild:
"hasChild"},
allowDragAndDrop: true
}); | Property: allowDragAndDrop

var treeObj
= new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text:
"name", hasChildren: "hasChild"},
allowDragAndDrop:
true
});
treeObj.appendTo("#tree"); |
```

```
| Prevent Drag and drop to another Treeview | Property:
allowDragAndDropAcrossControl

$("#tree").ejTreeView({
fields: {dataSource: treeData,
id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},
allowDragAndDrop:
true,
allowDragAndDropAcrossControl: false
}); | Can be achieved using,

var
treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text:
"name", hasChildren: "hasChild"},
allowDragAndDrop: true,
nodeDragStop: function(args)
{
if (args.draggedParentNode.closest('.e-treeview') !== args.dropTarget.closest('.e-treeview'))
{
args.cancel = true;
}}
});
treeObj.appendTo("#tree"); |
```

```
| Prevent sibling drop | Property: allowDropSibling

$("#tree").ejTreeView({
fields:
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild:
"hasChild"},
allowDragAndDrop: true,
allowDropSibling: false
}); | Can be achieved
using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChildren: "hasChild"},
nodeDragStop: function(args)
{
if(args.dropIndicator === "e-drop-next") {
args.cancel =
true;

}}
treeObj.appendTo("#tree"); |
```



| Prevent child drop | **Property:** `allowDropChild`  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, allowDragAndDrop: true, allowDropChild: false});` | **Can be achieved using,**  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, nodeDragStop: function(args) {if(args.dropIndicator === "e-drop-in") {args.cancel = true;}}});` `treeObj.appendTo("#tree");` |

| Triggers when node is dragged | **Event:** `nodeDrag`  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, allowDragAndDrop: true, nodeDrag: function() {}});` | **Event:** `nodeDragging`  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, allowDragAndDrop: true, nodeDragging: function() {}});` `treeObj.appendTo("#tree");` |

| Triggers when node drag is started successfully | **Event:** `nodeDragStart`  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, allowDragAndDrop: true, nodeDragStart: function() {}});` | **Event:** `nodeDragStart`  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, allowDragAndDrop: true, nodeDragStart: function() {}});` `treeObj.appendTo("#tree");` |

| Triggers before dragged node drag is stopped | **Event:** `nodeDragStop`  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, allowDragAndDrop: true, nodeDragStop: function() {}});` | **Event:** `nodeDragStop`  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, allowDragAndDrop: true, nodeDragStop: function() {}});` `treeObj.appendTo("#tree");` |

| Triggers when node is dropped successfully | **Event:** `nodeDropped`  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, allowDragAndDrop: true, nodeDropped: function() {}});` | **Event:** `nodeDropped`  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, allowDragAndDrop: true, nodeDropped: function() {}});` `treeObj.appendTo("#tree");` |

### Expand/Collapse nodes

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers before node is collapsed | **Event:** `beforeCollapse`  
`$("#tree").ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}, beforeCollapse: function() {}});` | **Event:** `nodeCollapsing`  
`var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}, nodeCollapsing: function() {}});` `treeObj.appendTo("#tree");` |

| Triggers before node is expanded | **Event:** *beforeExpand*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 beforeExpand: function() {}  
 | **Event:** *nodeExpanding*  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
 nodeExpanding: function() {}  
 treeObj.appendTo("#tree"); |

| Collapse all nodes | **Method:** *collapseAll*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 var treeObj = \$("#tree").data("ejTreeView");  
 treeObj.collapseAll(); | **Method:** *collapseAll*  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
 treeObj.appendTo("#tree");  
 treeObj.collapseAll(); |

| Collapse Node | **Method:** *collapseNode*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 var treeObj = \$("#tree").data("ejTreeView");  
 treeObj.collapseNode(\$("#1")); | **Method:** *collapseAll*  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
 treeObj.appendTo("#tree");  
 treeObj.collapseAll(["1"]); |

| Prevent multiple nodes expand | **Property:** *enableMultipleExpand*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 enableMultipleExpand: false  
 | **Can be achieved using,**  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
 nodeExpanding: function(args) {  
 var parent=args.node.parentNode.closest('.e-list-item');  
 if (parent === null)  
 parent=args.node.parentNode.closest('.e-treeview');  
 var children=parent.querySelectorAll('.e-list-item');  
 var i=0;  
 nodes=[];  
 while(i<children.length){  
 nodes.push(children[i].getAttribute("data-uid"));  
 i++;  
 }  
 this.collapseAll(nodes)  
 }  
 treeObj.appendTo("#tree"); |

| Expand all Nodes | **Method:** *expandAll*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 var treeObj = \$("#tree").data("ejTreeView");  
 treeObj.expandAll(); | **Method:** *expandAll*  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
 treeObj.appendTo("#tree");  
 treeObj.expandAll(); |

| Expand Node | **Method:** *expandNode*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 var treeObj = \$("#tree").data("ejTreeView");  
 treeObj.expandNode(\$("#1")); | **Method:** *expandAll*  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
 treeObj.appendTo("#tree");  
 treeObj.expandAll(["1"]); |

| Gets/Sets Expanded nodes | **Property:** *expandedNodes*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 expandedNodes: [1, 2]  
 | **Property:** *expandedNodes*  
 var treeObj = new ej.navigations.TreeView({dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},  
 expandedNodes: ["1", "2"]  
 treeObj.appendTo("#tree"); |

| Expand action | **Property:** *expandOn*  
 {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},  
 expandOn: "click"  
 | **Property:** *expandOn*  
 var treeObj = new ej.navigations.TreeView({dataSource:

treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>expandOn:  
"Click"<br/>});<br/>treeObj.appendTo("#tree"); |

| Get expanded nodes | **Method:** *getExpandedNodes*<br/><br/>\$("#tree").ejTreeView({<br/>fields:  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}<br/>});<br/>var  
treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.getExpandedNodes(); | **Can be achieved  
using**,<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id",  
parentId: "pid", text: "name", hasChildren: "hasChild"}<br/>});<br/>treeObj.appendTo("#tree");<br/>var  
expand=treeObj.expandedNodes;<br/>var i=0;<br/>var expandedNodes;<br/>while(i<expand.length)  
{<br/>expandedNodes.push(treeObj.element.querySelector('[data-uid="'+ expandednodes[i] +  
"']));<br/>i++;<br/>}<br/>console.log(expandedNodes) |

| Get expanded nodes index | **Method:**  
*getExpandedNodesIndex*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id",  
parentId: "pid", text: "name", hasChild: "hasChild"}<br/>});<br/>var treeObj =  
\$("#tree").data("ejTreeView");<br/>treeObj.getExpandedNodesIndex(); | Not Applicable |

| To check if node is expanded | **Method:** *isExpanded*<br/><br/>\$("#tree").ejTreeView({<br/>fields:  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}<br/>});<br/>var  
treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.isExpanded(\$("#book")); | **Can be achieved  
using**,<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id",  
parentId: "pid", text: "name", hasChildren:  
"hasChild"}<br/>});<br/>treeObj.appendTo("#tree");<br/>if(treeObj.expandedNodes.indexOf("book")  
!== -1) {<br/>console.log("Node is expanded")<br/>} |

| Triggers when node is collapsed successfully | **Event:**  
*nodeCollapse*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId:  
"pid", text: "name", hasChild: "hasChild"},<br/>nodeCollapse: function() {}<br/>}); | **Event:**  
*nodeCollapsed*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData,  
id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>nodeCollapsed: function()  
{<br/>});<br/>treeObj.appendTo("#tree"); |

| Triggers when node is expanded successfully | **Event:**  
*nodeExpand*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId:  
"pid", text: "name", hasChild: "hasChild"},<br/>nodeExpand: function() {}<br/>}); | **Event:**  
*nodeExpanded*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData,  
id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>nodeExpanded: function()  
{<br/>});<br/>treeObj.appendTo("#tree"); |

## Node Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Editing | **Property:** *allowEditing*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData,  
id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>allowEditing: true<br/>}); | **Property:**  
*allowEditing*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData,  
id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>allowEditing:  
true<br/>});<br/>treeObj.appendTo("#tree"); |

| Triggers before node is edited | **Event:** *beforeEdit*<br/><br/>\$("#tree").ejTreeView({<br/>fields:  
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>allowEditing:  
true,<br/>beforeEdit: function() {}<br/>}); | **Event:** *nodeEditing*<br/><br/>var treeObj = new

```
ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name",
hasChildren: "hasChild"},
allowEditing: true,
nodeEditing: function()
{}
});
treeObj.appendTo("#tree"); |
```

| To Enable editing programatically | Not Applicable | **Method:** *beginEdit*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>});<br/>treeObj.appendTo("#tree");<br/>treeObj.beginEdit("1"); |

| Triggers before node edit is successful | **Event:** *inlineEditValidation*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>allowEditing: true,<br/>inlineEditValidation: function() {}<br/>}); | **Event:** *nodeEdited*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>allowEditing: true,<br/>nodeEdited: function() {}<br/>});<br/>treeObj.appendTo("#tree"); |

| Triggers when node is edited successfully | **Event:** *nodeEdit*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>allowEditing: true,<br/>nodeEdit: function() {}<br/>}); | **Event:** *dataSourceChanged*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>dataSourceChanged: function() {}<br/>});<br/>treeObj.appendTo("#tree"); |

## Node Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Multi-selection | **Property:** *allowMultiSelection*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>allowMultiSelection: true<br/>}); | **Property:** *allowMultiSelection*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>allowMultiSelection: true<br/>});<br/>treeObj.appendTo("#tree"); |

| Triggers before node is selected | **Event:** *beforeSelect*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>beforeSelect: function() {}<br/>}); | **Event:** *nodeSelecting*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>nodeSelecting: function() {}<br/>});<br/>treeObj.appendTo("#tree"); |

| Fullrowselection | **Property:** *fullRowSelect*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>fullRowSelect: true<br/>}); | **Property:** *fullRowSelect*<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},<br/>fullRowSelect: true<br/>});<br/>treeObj.appendTo("#tree"); |

| Get selected node | **Method:** *getSelectedNode*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"}<br/>});<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.getSelectedNode(); | **Can be achieved using**,<br/><br/>var treeObj = new ej.navigations.TreeView({<br/>fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"}<br/>});<br/>treeObj.appendTo("#tree");<br/>var select=treeObj.selectedNodes;<br/>var

```
selectedNode;
selectedNode.push(treeObj.element.querySelector('[data-uid="'+ selectedNode[i] +
"']));
console.log(selectedNode) |
```

| Get selected node index | **Method:**

```
getSelectedNodeIndex

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChild: "hasChild"
}};
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getSelectedNodeIndex(); | Can be achieved
using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChildren: "hasChild"
}};
treeObj.appendTo("#tree");
var
nodes=treeObj.element.querySelectorAll('.e-list-item')
var nodeIndex;
var i =
0;
while(i<nodes.length) {
if(nodes[i].classList.contains('e-active')) {nodeIndex =
i;
break;
}
i++
}
console.log(nodeIndex) |
```

| Get selected nodes | **Method:** *getSelectedNodes*<br/><br/>\$("#tree").ejTreeView({<br/>fields:

```
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild:
"hasChild"},
allowMultiSelection: true
}};
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getSelectedNodes(); | Can be achieved using,

var
treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text:
"name", hasChildren: "hasChild"
}};
treeObj.appendTo("#tree");
var
select=treeObj.selectedNodes;
var i=0;
var selectedNodes;
while(i<select.length)
{
selectedNodes.push(treeObj.element.querySelector('[data-uid="'+ selectedNodes[i] +
"']));
i++
}
console.log(selectedNodes) |
```

| Get selected nodes index | **Method:**

```
getSelectedNodesIndex

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChild: "hasChild"},
allowMultiSelection: true
}};
var
treeObj = $("#tree").data("ejTreeView");
treeObj.getSelectedNodesIndex(); | Can be achieved
using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChildren: "hasChild"},
allowMultiSelection:
true
}};
treeObj.appendTo("#tree");
var nodes=treeObj.element.querySelectorAll('.e-list-
item')
var nodeIndex;
var i = 0;
while(i<nodes.length)
{
if(nodes[i].classList.contains('e-active'))
{nodeIndex.push(i);
}
i++
}
console.log(nodeIndex) |
```

| To check if node is selected | **Method:** *isSelected*<br/><br/>\$("#tree").ejTreeView({<br/>fields:

```
{dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"
}};
var
treeObj = $("#tree").data("ejTreeView");
treeObj.isSelected($("#book")); | Can be achieved
using,

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData, id: "id",
parentId: "pid", text: "name", hasChildren: "hasChild"
}};
treeObj.appendTo("#tree");
if
(treeObj.selectedNodes.indexOf("book") !== -1) {
console.log("Node is selected")
}
 |
```

| Triggers when node is selected successfully | **Event:**

```
nodeSelect

$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid",
text: "name", hasChild: "hasChild"},
nodeSelect: function() {}
}}; | Event:
nodeSelected

var treeObj = new ej.navigations.TreeView({
fields: {dataSource: treeData,
id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},
nodeSelected: function(args)
{
if (args.action == "select") {}
}};
treeObj.appendTo("#tree"); |
```

| Select all nodes | **Method:** *selectAll*<br/><br/>\$("#tree").ejTreeView({<br/>fields: {dataSource:
treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},<br/>allowMultiSelection:
true<br/>}};<br/>var treeObj = \$("#tree").data("ejTreeView");<br/>treeObj.selectAll(); | **Can be**

**achieved using**,  

```
var treeObj = new ej.navigations.TreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChildren: "hasChild"
 },
 allowMultiSelection: true
});
treeObj.appendTo("#tree");
var nodes = treeObj.element.querySelectorAll('.e-list-item');
var selectednodes;
for (int i = 0; i < nodes.length; i++) {
 selectednodes.push(nodes[i].getAttribute('data-uid'));
}
treeObj.selectedNodes = selectednodes;
| Gets/Sets selected node | Property: selectedNode

$("#tree").ejTreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChild: "hasChild"
 },
 selectedNode: 1
});
| Property: selectedNodes

var treeObj = new ej.navigations.TreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChildren: "hasChild"
 },
 selectedNodes: ["1"]
});
treeObj.appendTo("#tree");
| Select node | Method: selectNode

$("#tree").ejTreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChild: "hasChild"
 }
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.selectNode($("#book"));
| Can be achieved using,

var treeObj = new ej.navigations.TreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChildren: "hasChild"
 }
});
treeObj.appendTo("#tree");
treeObj.selectedNodes=["book"]
| Gets/Sets selected nodes | Property: selectedNodes

$("#tree").ejTreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChild: "hasChild"
 },
 selectedNodes: [1, 2],
 allowMultiSelection: true
});
| Property: selectedNodes

var treeObj = new ej.navigations.TreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChildren: "hasChild"
 },
 selectedNodes: ["1", "2"],
 allowMultiSelection: true
});
treeObj.appendTo("#tree");
| Triggers when node is unselected successfully | Event: nodeUnselect

$("#tree").ejTreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChild: "hasChild"
 },
 nodeUnselect: function() {}
});
| Event: nodeSelected

var treeObj = new ej.navigations.TreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChildren: "hasChild"
 },
 nodeSelected: function(args) {
 if (args.action == "un-select") {}
 }
});
treeObj.appendTo("#tree");
| To unselect all nodes | Method: unselectAll

$("#tree").ejTreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChild: "hasChild"
 },
 allowMultiSelection: true
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.unselectAll();
| Can be achieved using,

var treeObj = new ej.navigations.TreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChildren: "hasChild"
 },
 allowMultiSelection: true
});
treeObj.appendTo("#tree");
treeObj.selectedNodes=[];
| To unselect node | Method: unselectNode

$("#tree").ejTreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChild: "hasChild"
 },
 allowMultiSelection: true
});
var treeObj = $("#tree").data("ejTreeView");
treeObj.unselectNode($("#book"));
| Can be achieved using,

var treeObj = new ej.navigations.TreeView({
 fields: {
 dataSource: treeData,
 id: "id",
 parentId: "pid",
 text: "name",
 hasChildren: "hasChild"
 },
 allowMultiSelection: true
});
treeObj.appendTo("#tree");
var selectedNodes=treeObj.selectedNodes.pop(book);
```

## Template

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |



| Custom template | **Property:** `template`  
`<#tree>$.ejTreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},</>template: "#templateData",</>});` | **Property:** `nodeTemplate`  
`<#tree>var treeObj = new ej.navigations.TreeView({fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChildren: "hasChild"},</>nodeTemplate: "#templateData",</>});</>treeObj.appendTo("#tree");` |

## Uploader

### Async in EJ2 JavaScript Uploader control

The uploader component allows you to upload the files asynchronously. The upload process requires save and remove action URL to manage the upload process in the server.

- The save action is necessary to handle the upload operation
- The remove action is optional, can handle the removed files from server

The name attribute must match the name of a parameter in the POST method. For more information, refer [hear](#). The name attribute is automatically generated from the control's ID property. If the name attribute to be different from the ID property, then you can use the `htmlAttributes` property to set the name attribute directly

to the input element. For more information refer [hear](#).

The File can be uploaded automatically or manually. For more information, you can refer to the [Auto Upload](#) section from the documentation.

### Multiple file upload

By Default, the uploader component allows you to select and upload multiple files simultaneously. The selected files are organized in a list for every file selection until you clear it by clicking clear button that is shown in footer. You can add the multiple attributes to original input element of file by enabling the multiple file selection. The following example explains about [multiple](#) file upload settings.

In the following example, explains about multiple file upload settings.

#### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Uploader Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008c00;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```



### Single file upload

You can select and upload a single file by disabling the multiple file selection property. The file list item is removed for every selection and it always maintain a single file to upload. You can remove the multiple attributes from the original input element of file by enabling the single file upload property.

The following example explains about single file upload settings.

#### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 multiple: false
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
```

```

}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### Save Action

The save action handler upload the files that needs to be specified in the [saveUrl](#) property.

The save handler receives the submitted files and manages the save process in server. After uploading the files to server location, the color of the selected file name changes to green and the remove icon is changed as bin icon.

- When the file is uploaded successfully, the event “success” triggers to handle the operation after upload.
- When the file is failed to upload, the event “failure” triggers with information, which cause this failure.

You can cancel the upload process by setting the upload event argument **eventargs.cancel** to true.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save'
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Uploader Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### *Server-side configuration for save action*

Here's how to handle the server-side action for saving the file in server.

```
`c#
[AcceptVerbs("Post")]
public void Save()
{
 try
 {
 if (HttpContext.Current.Request.Files.AllKeys.Length > 0)
 {
 var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
 if (httpPostedFile != null)
 {
 var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
 var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
 if (!File.Exists(fileSavePath))
 {
 httpPostedFile.SaveAs(fileSavePath);
 HttpResponseMessage Response = HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusDescription = "File uploaded succesfully";
 Response.End();
 }
 }
 else
 {
 HttpResponseMessage Response = HttpContext.Current.Response;
 Response.Clear();
 Response.Status = "400 File already exists";
 Response.StatusCode = 400;
 Response.StatusDescription = "File already exists";
 Response.End();
 }
 }
 }
}
```

```

}
catch (Exception e)
{
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 400;
 Response.Status = "400 No Content";
 Response.StatusDescription = e.Message;
 Response.End();
}
}
`

```

### Remove Action

The remove action is optional. Specify the URL to handle remove process from server. The remove handler receives the posted files and handle the remove operation in server.

- When the files are removed successfully from server, the success event triggers to denote the process has completed.
- When remove action fails, the event “failure” triggers with information, which cause failure in remove process.

You can differentiate the file operation whether the success event triggers from save or remove action in its arguments **eventArgs.operation**.

You can remove the files which is not uploaded locally by clicking the remove icon. In this case, the success or failure events will not be triggered.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');

```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 Uploader</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Uploader Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

## STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue','calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

*Server-side configuration for remove action*

Here's how to handle the server-side action for removing the file from server.

```
`c#
[AcceptVerbs("Post")]
public void Remove()
{
 try
 {
 var fileSave = "";
 if (HttpContext.Current.Request.Form["cancel-uploading"] != null)
 {
 fileSave = HttpContext.Current.Server.MapPath("UploadingFiles");
 }
 else
 {
 fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
 }
 var fileName = HttpContext.Current.Request.Files["UploadFiles"].FileName;
 var fileSavePath = Path.Combine(fileSave, fileName);
 if (File.Exists(fileSavePath))
 {
 File.Delete(fileSavePath);
 }
 }
 catch (Exception e)
 {
 HttpResponseMessage Response = HttpContext.Current.Response;
 Response.Clear();
 Response.Status = "404 File not found";
 Response.StatusCode = 404;
 Response.StatusDescription = "File not found";
 Response.End();
 }
}
```

```

}
,

```

### Auto Upload

By default, the uploader processes the files to upload once the files are selected and added in upload queue. To upload manually, disable the [autoUpload](#) property. When you disable this property, you can use the action buttons to call upload all or clear all actions manually. You can change those buttons text using the [buttons](#) property in the Uploader component.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 // disable the auto upload functionalities
 autoUpload: false
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}

```



```

 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### Sequential Upload

By default, the uploader control process multiple files to upload simultaneously. When you enable the [sequentialUpload](#) property, the selected files will process sequentially (one after the other) to the server. If the file uploaded successfully or failed, the next file will upload automatically in this sequential upload. This feature helps to reduce the upload traffic and reduce the failure of file upload.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 // enable the sequential upload functionality
 sequentialUpload: true
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### Preload Files

The uploader component allows you to preload the list of files that are uploaded in the server. The preloaded files are useful to view and remove the files from server that can be achieved by the [files](#) property. By default, the files are configured with uploaded successfully state on rendering file list. By default, the files are configured with uploaded successfully state on rendering file list. The following properties are mandatory to configure the preloaded files:

- Name
- Size
- Type

## INDEX.TS

```
import { Uploader, FilesPropModel } from '@syncfusion/ej2-inputs';
let preLoadFiles: FilesPropModel[] = [
 {name: 'Books', size: 500, type: '.png'},
 {name: 'Movies', size: 12000, type: '.pdf'},
 {name: 'Study materials', size: 500000, type: '.docx'},
];
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 files: preLoadFiles
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
```

```

if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### Adding additional HTTP headers with upload action

The Uploader component allows you to add the additional headers with **save** and **remove** action request using **uploading** and **removing** event, which helps to send validation token on file upload. Access the current request and set the request header within these events.

The following code block shows how to add the additional headers with save and remove action request.

```

<div style='margin: auto 30%'>
<input type='file' id='uploader' name="UploadFiles"/>
</div>
`ts
import { Uploader } from '@syncfusion/ej2-inputs';
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 uploading: addHeaders,

```

removing: addHeaders

});

uploadObj.appendTo('#uploader')

function addHeaders(args: any) {

args.currentRequest.setRequestHeader('custom-header', 'Syncfusion');

}

,

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [How to add additional data on upload](#)
- [How to add confirm dialog to remove the files](#)
- [Check the MIME type of file before uploading it](#)
- [How to open and edit the uploaded files](#)

## Chunk upload in EJ2 JavaScript Uploader control

The Uploader sends the large file split into small chunks and transmits to the server using AJAX. You can also pause, resume, and retry the failed chunk file.

\* The chunk upload works in asynchronous upload only.

- This feature is available from the Essential Studio Vol 2, 2018 release.

To enable the chunk upload, set the size to [chunkSize](#) option of the upload and it receives the value in **bytes**.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove',
 // split the files into chunks of the size 102400 bytes
 chunkSize: 102400
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```
<title>Essential JS 2 Uploader</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Uploader Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" class="fileupload">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## **STYLES.CSS**

```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue','calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
```

The chunk upload functionality separates the selected files into blobs of the data or chunks. These chunks are transmitted to the server using an AJAX request. The chunks are sent in **sequential** order, and the next chunk can be sent to the server according to the [success](#) of the previous chunk. If any one of the chunk failed, then the remaining chunk cannot be sent to the server.

The [chunkSuccess](#) or [chunkFailure](#) event will be triggered when the chunk is sent to the server successfully or failed. If all the chunks are sent to the server successfully, the uploader [success](#) event is triggered.

Chunk upload will work when the selected file size is greater than the specified chunk size. otherwise, it upload the files normally.

### Additional configurations

To modify the chunk upload, the following options can be used.

- **RetryAfterDelay** - If error occurs while sending any chunk request from JavaScript, hold the operation for 500 milliseconds (by default), and retry the operation using chunk. This can be achieved by using the [asyncSettings.retryAfterDelay](#) property.

You can modify the holding time interval in milliseconds.

- **RetryCount** - Specifies the number of retry actions performed when the file fails to upload. By default, [retry](#) action is performed 3 times.

If the file fails to upload continuously, the request is aborted and the uploader [failure](#) event will trigger.

The following sample specifies the chunk upload delay with 3000 milliseconds and the retry count is 5. The failure event is triggered as the wrong saveUrl is used.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 // provided the wrong url to showcase the chunk upload failure
 // related properties.
 saveUrl:
 'https://aspnetmvc.syncfusion.com/service/api/uploadbox/Save',
 removeUrl:
 'https://aspnetmvc.syncfusion.com/service/api/uploadbox/Remove',
 // set chunk size for enable the chunk upload
 chunkSize: 102400,
 // set count for automatic retry when chunk upload failed
 retryCount: 5,
 // set time delay for automatic retry when chunk upload failed
 retryAfterDelay: 3000
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" class="fileupload">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## STYLES.CSS

```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
```



### Resumable upload

Allows you to resume an upload operation after a network failure or manually interrupts (pause) the upload. You can perform pause and resume upload actions using public methods ([pause](#) and [resume](#)) and UI interaction. The pause icon is enabled after the upload begins.

This pause and resume features available only when the chunk upload is enabled.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader ({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove',
 // set chunk size for enable the chunk upload
 chunkSize: 102400
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" class="fileupload">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
```

```

 ele.style.visibility = "visible";
 }
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### Cancel upload

The uploader component allows you to cancel the uploading file. This can be achieved by clicking the cancel icon or using the [cancel](#) method. The [canceling](#) event will be fired whenever the file upload request is canceled. While canceling the upload request, the partially uploaded file is removed from the server.

When the request fails, the pause icon is changed to retry icon. By clicking the retry icon, sends the failed chunk request again to the server and upload started from where it is failed. You can retry the canceled upload request again using retry UI or [retry](#) methods. But, if you retry this, the file upload action again starts from initial.

The following example explains about chunk upload with cancel support.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Remove',
 // set chunk size for enable the chunk upload
 chunkSize: 102400
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');

```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" class="fileupload">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

## STYLES.CSS

```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue','calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
```

The retry action has different working behavior for chunk upload and default upload.

- Chunk upload - Retries to upload the failed request where it is failed previously.
- Default upload - Retries to upload the failed file again from initial.

#### Server-Side configurations

The server-side implementation entirely depends on the application requirements and logic. The following code snippet provides the server-side logic to handle the chunk upload using the uploader components.

```
`c#
// Server configuration for upload a file.
public void Save()
{
 try
 {
 if (HttpContext.Current.Request.Files.AllKeys.Length > 0)
 {
 var httpPostedChunkFile = HttpContext.Current.Request.Files["chunkFile"];
 if (httpPostedChunkFile != null)
 {
 var saveFile = HttpContext.Current.Server.MapPath("UploadingFiles");
 // Save the chunk file in temporary location with .part extension
 var SaveFilePath = Path.Combine(saveFile, httpPostedChunkFile.FileName + ".part");
 var chunkIndex = HttpContext.Current.Request.Form["chunkIndex"];
 if (chunkIndex == "0")
 {
 httpPostedChunkFile.SaveAs(SaveFilePath);
 }
 else
 {
 // Merge the current chunk file with previous uploaded chunk files
 MergeChunkFile(SaveFilePath, httpPostedChunkFile.InputStream);
 var totalChunk = HttpContext.Current.Request.Form["totalChunk"];
 if (Convert.ToInt32(chunkIndex) == (Convert.ToInt32(totalChunk) - 1))
 {
```

```
var savedFile = HttpContext.Current.Server.MapPath("UploadedFiles");
var originalFilePath = Path.Combine(savedFile, httpPostedChunkFile.FileName);
// After all the chunk files completely uploaded, remove the .part extension and move this file into save
location
System.IO.File.Move(SaveFilePath, originalFilePath);
}
}
HttpResponse ChunkResponse = HttpContext.Current.Response;
ChunkResponse.Clear();
ChunkResponse.ContentType = "application/json; charset=utf-8";
ChunkResponse.StatusDescription = "File uploaded succesfully";
ChunkResponse.End();
}
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
if (httpPostedFile != null)
{
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
if (!File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
else
{
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.Status = "400 File already exists";
Response.StatusCode = 400;
```

```
Response.StatusDescription = "File already exists";
Response.End();
}
}
}
}
catch (Exception e)
{
 HttpResponseMessage Response = HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 400;
 Response.Status = "400 No Content";
 Response.StatusDescription = e.Message;
 Response.End();
}
}
// Server configuration for remove a uploaded file
public void Remove()
{
 try
 {
 var fileSave = "";
 if (HttpContext.Current.Request.Form["cancelUploading"] != null)
 {
 fileSave = HttpContext.Current.Server.MapPath("UploadingFiles");
 } else
 {
 fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
 }
 var fileName = HttpContext.Current.Request.Files["UploadFiles"].FileName;
 var fileSavePath = Path.Combine(fileSave, fileName);
 if (File.Exists(fileSavePath))
```

```
{
File.Delete(fileSavePath);
}
}
catch (Exception e)
{
HttpResponse Response = HttpContext.Current.Response;
Response.Clear();
Response.Status = "404 File not found";
Response.StatusCode = 404;
Response.StatusDescription = "File not found";
Response.End();
}
}
// Merge the current chunk file with previous uploaded chunk files
public void MergeChunkFile(string fullPath, Stream chunkContent)
{
try
{
using (FileStream stream = new FileStream(fullPath, FileMode.Append, FileAccess.Write,
FileShare.ReadWrite))
{
using (chunkContent)
{
chunkContent.CopyTo(stream);
}
}
}
catch (IOException ex)
{
throw ex;
}
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

## File source in EJ2 JavaScript Uploader control

### Paste to upload

The uploader component allows you to upload the files using the select or drop files option from the file explorer. It also supports pasting to upload the image files. You can upload any currently copied images in the clipboard.

When you paste the image, it will be saved in the server with the filename as `image.png`. The file name can be renamed in the server end. You can generate a random name for the file name using `getUniqueID` method. Refer to the following example.

### INDEX.TS

```
import { Uploader, UploadingEventArgs } from '@syncfusion/ej2-inputs';
import { getUniqueID } from '@syncfusion/ej2-base';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 autoUpload: false,
 uploading: onUploadBegin
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
function onUploadBegin(args: UploadingEventArgs): void {
 // check whether the file is uploading from paste.
 if (args.fileData.fileSource === 'paste') {
 let newName: string = getUniqueID(args.fileData.name.substring(0,
args.fileData.name.lastIndexOf('.'))) + '.png';
 args.customFormData = [{ 'fileName': newName }];
 }
}
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
```



```
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### STYLES.CSS

```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
```

*Save action for paste to upload*

`c#

```
public void Save() {
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
// Get the current file name
var oldName = httpPostedFile.FileName;
```

```
// Get the additional data as name in server end by corresponding key.
var newName = HttpContext.Current.Request.Form["fileName"];
// Rename the file
File.Move(oldName, newName);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
// Sending the file path to client side
Response.StatusDescription = fileSavePath;
Response.End();
}
}
,
```

### Directory upload

The uploader component allows you to upload all files in the folders to server by using the [directoryUpload](#) property. When this property is enabled, the uploader component processes the files by iterating through the files and sub-directories in a directory. It allows you to select only folders instead of files to upload.

The directory upload is available only in browsers that supports **HTML5 directory**. The uploader will process directory upload by dragging and dropping in the Edge browser. Refer to the following example to upload files to the server.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 directoryUpload: true
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
```

```

<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### **STYLES.CSS**

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

*Save action for directory upload*

```
`c#
```

```

public void Save() {
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];

```

```
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
// split the folders by using file name
string[] folders = httpPostedFile.FileName.Split('/');
string fileSavePath = "";
if (folders.Length > 1)
{
 for (var i = 0; i < folders.Length - 1; i++)
 {
 var newFolder = Path.Combine(fileSave, folders[i]);
 // create folder
 Directory.CreateDirectory(newFolder);
 fileSave = newFolder;
 }
 fileSavePath = Path.Combine(fileSave, folders[folders.Length - 1]);
}
else
{
 fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
}
if (!System.IO.File.Exists(fileSavePath))
{
 // save file in the corresponding server location
 httpPostedFile.SaveAs(fileSavePath);
 HttpResponse Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 // Sending the file path to client side
 Response.StatusDescription = fileSavePath;
 Response.End();
}
}
```

## Drag and drop

The uploader component allows you to drag and drop the files to upload. You can drag the files from file explorer and drop into the drop area. By default, the uploader component act as drop area element. The drop area gets highlighted when you drag the files over drop area.

## Custom drop area

The uploader component allows you to set external target element as drop area using the [dropArea](#) property. The element can be represented as HTML element or element's id.

## INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 autoUpload: false,
 dropArea: document.getElementById('droparea')
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" style="width: 90%">
 <div id="droparea">
 Drop files here to upload
 </div>
 <div id="uploadfile">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>
 </div>
</script>
var ele = document.getElementById('container');
```

```

if(ele) {
 ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### **STYLES.CSS**

```

#container {
 visibility: hidden;
 padding-left: 5%;
 width: 100%;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.fileupload {
 margin: 20px auto;
 width: 400px;
}
#droparea {
 padding: 50px 25px;
 margin: 30px auto;
 border: 1px solid #c3c3c3;
 text-align: center;
 width: 20%;
 display: inline-flex;
}
.e-file-select,
.e-file-drop {
 display: none;
}
body .e-upload-drag-hover {
 outline: 2px dashed brown;
}
#uploadfile {
 width: 60%;
 display: inline-flex;
 margin-left: 5%;
}

```

### *Customize drop area*

You can customize the appearance of drop area by overriding the default drop area styles. The class `""` and `""` is available to handle this customization.

### **INDEX.TS**

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 dropArea: document.getElementById('droparea')
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
document.getElementById('browse').onclick = function() {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
 return false;
}
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="dropArea" style="height: auto; overflow: auto">
 Drop files here or <a href=""
id="browse"><u>Browse</u>
 <input type="file" id="fileupload">
 </div>
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
```

```
}
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### STYLES.CSS

```
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.e-file-select-wrap {
 display: none;
}
#dropArea .e-upload {
 border: 0;
 margin-top: 15px;
}
#drop {
 padding-left: 30%;
}
#dropArea {
 min-height: 18px;
 border: 1px dashed #c3c3cc;
 padding-top: 15px;
 margin: 20px auto;
 width: 400px;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [Achieve file upload programmatically](#)
- [Validate image/\\* on drop](#)

### Validation in EJ2 JavaScript Uploader control

The uploader component validate the selected files size and extension using the [allowedExtensions](#), [minFileSize](#) and [maxFileSize](#) properties. The files can be validated before uploading to the server and can be ignored on uploading. Also, you can validate the files by setting the HTML attributes to the original input element. The validation process occurs on drag-and-drop the files also.



## File type

You can allow the specific files alone to upload using the [allowedExtensions](#) property. The extension can be represented as collection by comma separators. The uploader component filters the selected or dropped files to match against the specified file types and processes the upload operation. The validation happens when you specify value to inline attribute to accept the original input element.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 allowedExtensions: '.jpg,.png'
});
uploadObj.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
```

```

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### File size

The uploader component allows you to validate the files based on its size. The validation helps to restrict uploading large files or empty files to the server. The size can be represented in **bytes**. By default, the uploader component allows you to upload **minimum file size** as 0 byte and **maximum file size** as 28.4 MB using the [minFileSize](#) and [maxFileSize](#) properties.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
// Initialize the uploader component
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 minFileSize: 10000,
 maxFileSize: 1500000
});
uploadObj.appendTo('#fileupload');

```

### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

### Maximum files count

You can restrict uploading the maximum number of files using the **selected** event. In the selected event arguments, you can get the currently selected files details using the `getFilesData()`. You can modify the files details and assign the modified file list to the `eventArgs.modifiedFilesData`.

### INDEX.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { Uploader, FileInfo, SelectedEventArgs } from '@syncfusion/ej2-
inputs';
import { detach } from '@syncfusion/ej2-base';

```

```
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
popups';
enableRipple(true);
let dropElement: HTMLElement = document.getElementsByClassName('control-
fluid')[0] as HTMLElement;
// Initialize the control with file validation
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 minFileSize: 10000,
 allowedExtensions: '.doc, .docx, .xls, .xlsx',
 asyncSettings: {
 saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 selected: onFileSelected,
 success: onUploadSuccess,
 dropArea: dropElement
});
uploadObj.appendTo('#validation');
function onFileSelected(args : SelectedEventArgs) : void {
 // Filter the 5 files only to showcase
 args.filesData.splice(5);
 let filesData : FileInfo[] = uploadObj.getFilesData();
 let allFiles : FileInfo[] = filesData.concat(args.filesData);
 if (allFiles.length > 5) {
 for (let i : number = 0; i < allFiles.length; i++) {
 if (allFiles.length > 5) {
 allFiles.shift();
 }
 }
 args.filesData = allFiles;
 // set the modified custom data
 args.modifiedFilesData = args.filesData;
 }
 args.isModified = true;
}
function onUploadSuccess(args: any): void {
 let li: HTMLElement = this.uploadWrapper.querySelector('[data-file-
name="' + args.file.name + '"]');
 if (args.operation === 'upload') {
 (li.querySelector('.e-file-delete-btn') as HTMLElement).onclick
= () => {
 generateSpinner(this.uploadWrapper);
 };
 (li.querySelector('.e-file-delete-btn') as
HTMLElement).onkeydown = (e: any) => {
 if (e.keyCode === 13) {
 generateSpinner(e.target.closest('.e-upload'));
 }
 };
 } else {
 hideSpinner(this.uploadWrapper);
 detach(this.uploadWrapper.querySelector('.e-spinner-pane'));
 }
}
```

```
function generateSpinner(targetElement: HTMLElement): void {
 createSpinner({ target: targetElement, width: '25px' });
 showSpinner(targetElement);
}
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="validation" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## STYLES.CSS

```
#container {
 visibility: hidden;
}
#loader {
 color: #008c8c;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
```

```
top: 45%;
width: 30%;
}
```

### Duplicate files

You can validate the duplicate files before uploading to server using the selected event. Compare the selected files with the existing files data and filter the file list by removing the duplicate files.

#### INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { Uploader, FileInfo } from '@syncfusion/ej2-inputs';
import { detach, isNullOrUndefined } from '@syncfusion/ej2-base';
// Initialize the control with file validation
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 asyncSettings: {
 saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 selected: onFileSelect,
});
uploadObj.appendTo('#validation');
function onFileSelect(args : any) {
 let existingFiles: FileInfo[] = this.GetFilesData();
 for (let i: number = 0; i < args.filesData.length; i++) {
 for (let j: number = 0; j < existingFiles.length; j++) {
 if (!isNullOrUndefined(args.filesData[i])) {
 if (existingFiles[j].name == args.filesData[i].name) {
 args.filesData.splice(i, 1);
 }
 }
 }
 }
 existingFiles = existingFiles.concat(args.filesData);
 args.modifiedFilesData = existingFiles;
 args.isModified = true;
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
```

```
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="validation" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### STYLES.CSS

```
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [Validate image/\\* on drop](#)
- [Determine whether uploader has file input \(required validation\)](#)
- [Check file size before uploading it](#)
- [Check the MIME type of file before uploading it](#)

## Form support in EJ2 JavaScript Uploader control

The Uploader component works with HTML form like default file input. The following configuration is must to make the Uploader work inside the form.

- `saveUrl` and `removeUrl` must be null.
- `autoUpload` must be disabled.
- `name` attribute must be added in input element.

The selected or dropped files are received as a collection in form action when the form is submitted. The form action handles the server-side operations that manage the file upload process. When you reset the form, the file list and data will be cleared.

### INDEX.TS

```
import { FormValidator } from '@syncfusion/ej2-inputs';
import { Uploader } from '@syncfusion/ej2-inputs';
import { Dialog } from '@syncfusion/ej2-popups';
/**
 * Uploader form support sample
 */
let dropElement: HTMLElement = document.getElementsByClassName('control-fluid')[0] as HTMLElement;
// Initialize the uploader component
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 selected: onFileSelect,
 allowedExtensions: 'image/*',
 dropArea: dropElement
});
uploadObj.appendTo('#fileupload');
document.getElementById('browse').onclick = () => {
 document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button').click(); return false;
};
function onFileSelect(args: any): void {
 let inputElement: HTMLInputElement = document.getElementById('upload') as HTMLInputElement;
 inputElement.value = args.filesData[0].name;
}
let options: any = {
 // Initialize the CustomPlacement.
 customPlacement: (inputElement: Element, errorElement: Element) => {
 inputElement = inputElement.closest('.form-group').querySelector('.error');
 inputElement.parentElement.appendChild(errorElement);
 },
 rules: {
 'Name': {
 required: true
 },
 'Email': {
 required: true
 },
 'upload': {
 required: true
 }
 }
}
```



```

 }
 }
};
let formObj: FormValidator = new FormValidator('#form1', options);
function onFormSubmit(): void {
 let formStatus: Boolean = formObj.validate();
 if (formStatus) {
 formObj.element.reset();
 confirm.show();
 }
}
let confirm: Dialog = new Dialog({
 width: '335px',
 visible: false,
 content: 'Your details has been updated successfully, Thank you',
 target: document.getElementById('control_wrapper'),
 isModal: true,
 animationSettings: {
 effect: 'Zoom'
 }
});
confirm.appendTo('#confirmationDialog');
document.getElementById('submit-btn').onclick = () => {
 onFormSubmit();
};
confirm.overlayClick = () => {
 confirm.hide();
};
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 UPloader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Slider Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">

```

```

<div class="col-lg-12 control-section">
 <h4 class="form-title">Photo Contest</h4>
 <div class="control_wrapper" id="control_wrapper">
 <!-- Initialize Uploader -->
 <form id="form1" method="post">
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <input type="text" id="name" name="Name" data-
required-message="* Enter your name" required="" data-msg-
containerid="nameError">

 <label class="e-float-text e-label-top"
for="name">Name</label>
 </div>
 <div id="nameError"></div>
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <input type="email" id="email" name="Email"
data-validation="email" data-required-message="* Enter your email"
required="" data-msg-containerid="mailError">

 <label class="e-float-text e-label-top"
for="email">Email</label>
 </div>
 <div id="mailError"></div>
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <input type="text" id="mobilen0"
name="MobileNo" data-required-message="* Enter your mobile number"
required="" data-msg-containerid="noError">

 <label class="e-float-text e-label-top"
for="mobilen0">Mobile No</label>
 </div>
 <div id="noError"></div>
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input upload-area">
 <input type="text" id="upload" name="upload"
readonly="" data-required-message="* Select any file" required="" data-msg-
containerid="uploadError">

 <label class="e-float-text e-label-top"
for="upload">Choose a file</label>
 </div>
 <button id="browse" class="e-control e-btn e-
info">Browse..</button>
 <div id="uploadError"></div>
 <input type="file" name="UploadFiles"
id="fileupload">
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <textarea class="address-field" rows="4"
id="Address" name="Address"></textarea>

```

```


 <label class="e-float-text e-label-top"
for="address">Address</label>
 </div>
 </div>
</form>
<div class="submitBtn">
 <button class="submit-btn e-btn" id="submit-
btn">Submit</button>
</div>
<div id="confirmationDialog"></div>
</div>
</div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

.form-group #upload {
 width: 70%;
}
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
/* csslint ignore:start */
.e-float-input.upload-area {
 width: 69%;
}
/* csslint ignore:end */
.address-field {
 max-height: 50px;
}
#control_wrapper {
 max-width: 440px;
 margin: auto;
 border: 0.5px solid grey;
 padding: 1% 4% 2%;
}

```

```

 background: #f9f9f9;
 }
 .highcontrast #control_wrapper {
 background: #000000;
 }
 .upload-area, .e-bigger .upload-area {
 width: 69%;
 }
 .e-error {
 padding-top: 3px;
 }
 .e-upload {
 width: 100%;
 position: relative;
 margin-top: 15px;
 }
 .control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
 }
 .submit-btn {
 margin-top: 15px;
 position: relative;
 }
 .submitBtn {
 position: relative;
 left: 79%;
 }
 .form-support {
 width: 100%;
 }
 .success .form-support {
 display: none;
 }
 .success .successmsg {
 border: 0.5px solid green;
 padding: 10%;
 color: green;
 }
 #form1 {
 position: relative;
 top: 14%;
 }
 .form-support td {
 width: 100%;
 padding-top: 4%;
 }
 .e-upload {
 display: none;
 }
 .choose-file{
 width: 60%;
 }
 #browse {
 left: 46%;
 float: right;
 margin-top: 20px
 }

```

```
.e-bigger #browse {
 top: -41px;
}
/* csslint ignore:start */
.e-bigger.material #browse {
 top: -35px;
}
.e-bigger.fabric #browse, .e-bigger.highcontrast #browse {
 top: -41px;
 left: 50%;
}
/* csslint ignore:end */
.fabric #browse, .highcontrast #browse {
 top: -32px;
}
.bootstrap #browse {
 top: -34px;
}
.form-title {
 text-align: center;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

## Template in EJ2 JavaScript Uploader control

You can customize the default appearance of uploader using a template along with buttons.

### File list template

The [template](#) property is used to customize the default appearance of each file in the list. It can be represented as the HTML element or string. The selected or dropped files are displayed as per the template layout provided. The remove and progress bar action is handled using the corresponding events when the template is defined.

For example, you can display file type icon along with default UI elements.

### INDEX.TS

```
import { Uploader, FileInfo, SelectedEventArgs } from '@syncfusion/ej2-
inputs';
import { Event, detach } from '@syncfusion/ej2-base';
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 template: "<span class='icon sf-icon-
${type}'>${name}" +
 "${size} bytes
 " +
 "<progress id='progressBar' class='progressbar' value='0'
max='100'></progress> ",
 dropArea: document.getElementById('dropArea'),
 progress: onFileUpload,
```

```

 selected: onSelect,
 success: onuploadSuccess,
 failure: onuploadFailed
 });
 uploadObj.appendTo('#fileupload');
 document.getElementById('browse').onclick = function() {
 document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button').click();
 return false;
 }
 document.getElementById('clearbtn').onclick = () => {
 uploadObj.element.value = '';
 detach(document.getElementById('dropArea').querySelector('.e-upload-files'));
 };
 function onFileUpload(args: any) {
 let li: HTMLElement = this.uploadWrapper.querySelector('[data-file-name="' + args.file.name + '"]');
 let progressValue: number = Math.round((args.e.loaded / args.e.total) * 100);
 li.getElementsByTagName('progress')[0].value = progressValue;
 li.getElementsByClassName('percent')[0].textContent = progressValue.toString() + " %";
 }
 function onuploadSuccess(args: any) {
 if (args.operation === 'remove') {
 let height: string =
 document.getElementById('dropArea').style.height;
 height = (parseInt(height) - 40) + 'px';
 document.getElementById('dropArea').style.height = height;
 } else {
 let li: HTMLElement = this.uploadWrapper.querySelector('[data-file-name="' + args.file.name + '"]');
 let progressBar: HTMLElement =
 li.getElementsByTagName('progress')[0];
 progressBar.classList.add('e-upload-success');
 li.getElementsByClassName('percent')[0].classList.add('e-upload-success');
 let height: string =
 document.getElementById('dropArea').style.height;
 document.getElementById('dropArea').style.height = parseInt(height) - 15 + 'px';
 }
 }
 function onuploadFailed(args: any) {
 let li: HTMLElement = this.uploadWrapper.querySelector('[data-file-name="' + args.file.name + '"]');
 let progressBar: HTMLElement = li.getElementsByTagName('progress')[0];
 progressBar.classList.add('e-upload-failed');
 li.getElementsByClassName('percent')[0].classList.add('e-upload-failed');
 }
 function onSelect(args: SelectedEventArgs) {
 let length: number = args.filesData.length;
 let height: string = document.getElementById('dropArea').style.height;
 height = parseInt(height) + (length * 55) + 'px';
 document.getElementById('dropArea').style.height = height;
 }

```

```

}
document.getElementById('dropArea')!.onclick = (e: any) => {
 let target: HTMLElement = <HTMLElement>e.target;
 if (target.classList.contains('e-file-delete-btn')) {
 for (let i: number = 0; i < uploadObj.GetFilesData().length; i++) {
 if (
 (target.closest('li') as HTMLLIElement).getAttribute(
 'data-file-name'
) === uploadObj.GetFilesData()[i].name
) {
 uploadObj.remove(uploadObj.GetFilesData()[i]);
 }
 }
 } else if (target.classList.contains('e-file-remove-btn')) {
 detach(target.closest('li') as HTMLLIElement);
 }
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="dropArea">
 Drop files here or <a href=""
id="browse"><u>Browse</u>
 <input type="file" id="fileupload">
 </div>
 <div style="margin-left: 50px; padding-top:25px;">
 <button class="e-btn e-css" id="clearbtn">Clear All</button>
 </div>
 </div>
<script>
var ele = document.getElementById('container');

```

```

if(ele) {
 ele.style.visibility = "visible";
}

</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.control_wrapper {
 max-width: 500px;
 margin: auto;
}
.e-upload {
 width: 100%;
}
.e-file-select-wrap {
 display: none;
}
.e-upload {
 border: none;
 margin-top: 15px;
}
#drop {
 padding-left: 30%;
 font-size: 14px;
 font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif"
}
#dropArea {
 min-height: 18px;
 border: 1px dashed #c3c3cc;
 padding: 15px 0;
 margin: 0 auto;
 width: 440px;
}
.wrapper{
 padding: 0 10px;
}
.e-upload-success, .e-upload-failed {
 display: none;
}

```



```

}
.sf-icon-bmp:before { content: "\e700"; }
.sf-icon-xlsx:before, .sf-icon-XLSX:before { content: "\e701"; }
.sf-icon-avi:before, .sf-icon-AVI:before { content: "\e702"; }
.sf-icon-doc:before, .sf-icon-DOC:before { content: "\e703"; }
.sf-icon-exe:before { content: "\e704"; }
.sf-icon-mp4:before { content: "\e705"; }
.sf-icon-zip:before, .sf-icon-ZIP:before { content: "\e706"; }
.sf-icon-tar:before { content: "\e707"; }
.sf-icon-docx:before { content: "\e708"; }
.sf-icon-jpg:before { content: "\e709"; }
.sf-icon-png:before { content: "\e70a"; }
.sf-icon-gif:before { content: "\e70b"; }
.sf-icon-pdf:before { content: "\e70c"; }
.sf-icon-txt:before { content: "\e70d"; }
.sf-icon-jpeg:before { content: "\e70e"; }
.sf-icon-xls:before { content: "\e70f"; }
.sf-icon-mp3:before { content: "\e710"; }
#dropArea .e-upload .e-upload-files .e-file-delete-btn,
#dropArea .e-upload .e-upload-files .e-file-remove-btn {
 top: 35%;
}
span.file-icon{
 width: 25px;
 height: 25px;
 display: inline-flex;
 background-size: contain;
 margin: 7px;
}
img.file-icon {
 width: 20px;
 height: 20px;
}
.file-name {
 color: #e1e1e1;
 font-size: 14px;
 padding: 3px 10px;
 overflow: hidden;
 text-overflow: ellipsis;
 display: inline-block;
 width: 50%;
 white-space: nowrap;
 position: relative;
 top: 5px;
}
.file-size {
 font-size: 12px;
 padding: 3px 10px;
 overflow: hidden;
 display: inline-block;
 position: relative;
 top: 6px;
}
}
.progressbar{
 height: 5px;
 width: 70%;
 margin-left: 14px;

```

```
.upload-success{
 background-color: green
}
.upload-failed{
 background-color: red;
}
#dropArea .e-upload .e-upload-files .e-upload-file-list {
 min-height: 63px;
}
#dropArea .e-upload {
 border-width: 0 1px;
}
/* csslint ignore:start */
@font-face {
font-family: 'FileType_Font';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1tSgIAAAEoAAAAVmNtYXNDYOfNAAABYAAAAFZnbHlmBIu
kdGAAAKgAAB78aGVhZA/Zy4MAAADQAAAAANmhoZWEH1AQTAARAAACRobXR4SAAAAAAAAAYAAAB
IbG9jYUgKP2YAAAIgAAAAJm1heHABKQFJAAABCAAAACBuYw1lPdPGuQAIIUQAAAJtcG9zdGQHRBs
AACO0AAAAkAABAAAEAAAAAFwEAAAAAADdwABAAAAAAAAAAAAAAAAAAAAAAEgABAAAAQAavWuGj18
PPPUACwQAAAAANaiQ0kAAAAA1qJDSQAAAAADdwP0AAAACAACAAAAAAAAAAAAEAAAASAT0ADQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQA2AABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnEAQAAAAAXAQAAAAAAAAABAAAAAAAAAAAAAAAAQAAAA
EAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQ
AAAAEAAAAAAGAAAAAUAAMAAQAAABQABABCAAAABAAEAAEOcQ//8AAOcA//8AAAABAAQ
AAAABAAIAAwAEAAUABgAHAAGACQAKAAsADAANAA4ADwAQABEAAAAAAAAABCGHaAswDrAQ4BNQFsAZ
iB8wIsAlKCsgMAAyEDbgOrg9+AAAAACAAAAADdwP0ABIAJQA4AFgAZwCRANIA3QAAATM/Bj0BLwY
jJTM/Bj0BLwUrAQU7AT8GLwcjJRczHwsVDwsjFSM1Ixc3MxUjNTcHIycXFSM1IzMfChUPBh8HDwo
jNQMRFR8NMyEzPw01ESsBLwg9ASEjDw0FHwcZJwEjJQcGBgUDAwICAgMEBgYHJwGOJwGHBgUEAwI
CAwQFBGcCHKP5yIAGGBQUEAwEBAQEDAwUFBwgGAbUHCA0GBgUFBAQDAwIBAQBEBAGMDBAQFBQYMDy4
hvTc2KiEDNxc4BCFmCAcNCwoDBAMCAGEBAGIDBAQGBGcGBgQEAwEBAQIBAwMDBAUKDA5MegICAwQ
FBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBGsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgH
0AQEDBAUFBgbb+gE/AQEDAwQFBgYHBgUEAwMBAQsBAQMDBQUHBwcGBgUEAwI0AgIDBAQFBGcFBQM
DAgEBGgEDAWEBAQEByYFBgYHDQYFBgQFBAMDAwQCRryQkLw+U5GRUz68AQIEBgMEBAUFBQwBgY
FBQQEAWMDBQUGBGcIBwsFBQUEBAMFBAK8AbX81AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQk
JCgJSAQEFBwgKCwYHBvOCAGMEBQUGBwCICAKJCYCGBgUFBAICAFoAAAYAAAAA3cD9AAZADMATQB
ZAGQApQAAATMfAx0BDwMjISMvAz0BPwMzJTMfAx0BDwMrAi8DPQE/AzM3Mx8DHQEPaysCLwM9AT8
DMycXNzMhFyMnByM3JwEfBzMnBREVHw0zITM/DTURKEvCD0BIQ80AuoDAwUEAQEEBQMD/iwEAWQ
EAQEEBAMEADQDAwUEAQEEBQMD2gQDBAQEBQEAwTaAwMFBABEBAUDA9oEAWQEAEQEEBAME1SsJ0B
CJy4tKENBAWoBAQMEBQUGBtv6/gwCAGMEBQUGBwCICAKJCQoCMgoJCQkICAcHBgUFBAMCAvoGBwY
LCggHBQEB/qgKCQkJCAGHBwYFBQQDAgIBBgEEBQMDAwMFAwEBAwUDAwMDBQQBfQEEBQMDAwMFAwE
BAwUDAwMDBQQBfQEEBQMDAwMFAwEBAwUDAwMDBQQBCUREXF9GR19cARAGBgUFBAMBafpe/NQKCQk
JCAgHBwYFBQQDAgICAgMEBQUGBwCICAKJCQoCUgEBBQcICGsGBwb6AQECAwQFBQcGBwgICQkJAAA
ABQAAAAADdwP0AAIAQwCEAI8A0AAAAQc1BxUfDz8OPQEvDg8OBQcVDw4vDz8PHw4DHwcZJwURFR8
NMyEzPw01ESsBLwg9ASEPDgJefX0BAGQGBwkKCwwNDw8PERESERERDw8PDQwLCgkHBgQDAwQGBwk
KCwwNDw8PEREREHERDw8PDQwLCgkHBgQCAXYBAwUHCQoMDQ4QERETFBQUFRQUEhIREA4NDAoJBgU
DAgIDBQYJCgWnDhAREhIUFBUFFBQTEREQDg0MCgkHBQNDAQEDBAUFBgbb+v4MAgIDBAUFBGcHCAg
JCQkKAjIKCQkJCAGHBwYFBQQDAg16BgCGCwoIBWUBAf6oCgkJCQgIBwcGBQUEAwICAXNONU8JCBE
REA8ODQwLCgkHBgQDAQEDBAYHCQoLDA0ODxARERESERAQDw4ODAsKCAgFBQIBAQIFBQgICGsMDg4
PEBAREgoKFRMTEREQDg0MCgkHBQMBAQMFBwkKDA0OEBERExMVFBUUFBISEQ8PDQwKCAcFAwEBAwU
HCAoMDQ8PERISFBQBkQYGBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJ
SAQEFBwgKCwYHBvOCAGIDBAUFBwYHCAGJCQkAAAYAAAAA3cD9AAZADMATQBmaHEASgAAATMfAx0
BDwMjISMvAz0BPwMzJTMfAx0BDwMrAi8EPwQzNmFAx0BDwMrAi8EPwQzJR8CMz8CMx8BMz8BMwC
jLwIXDwEjJyUFBzMnBREVHw0zITM/DTURKEvCD0BIQ80AwoDAwUDAQEDBQMD/gwEAWQEAEQEEBAM
EAFQDAwUDAQEDBQMDvAMDBQMBAQEBAwUDA7wDAwUDAQEDBQMDvAMDBQMBAQEBAwUDA/7AJwECAQE
```

DKXsnAwEBKR04HycCAQIsHjkBjgEBAwQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgsKCACFAQH+qAoJCQkICAcHBgUFBAMCAGElAQMFawMEAgUEAQEEBQIEAwMFAwF9AQMFawMEAgUEAQEEBQIEAwMFAwF9AQMFawMEAgUEAQEEBQIEAwMFAwEBhgMNCAMfhhAHj7J/CAkHibL5BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAGJCQkKAlIBAUHCAoLBgcG+gEBAGMEBQUHBgcICAKJCQAAAAFAAAAAAN3A/QACwAXACMALgBvAAABFSVMxUjFTMVIZUjFzcZBxcjJwcjNycjFSVMxUjFTMVIZUBHwczJwURFR8NMyEzPw01ESsBLwg9ASEPDgLYWU1NWnuOKCcOTomKCKmOzkPWU1NWnsBWwEBawQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUAwIC+gYHBgsKCACFAQH+qAoJCQkICAcHBgUFBAMCAGHhGzMaOhq8RERdX0VFX10bMxo6GrwBOAYGBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAGICAwQFBQYHBwgICQkJCgJSAQEFBwgKCwYHBvOBAQIDBAUFBwYHCAGJCQkAAAAACAAAAADdwP0AAQACAAMABEAFQAxADwAfQAAARUzNSMFMzUjFzM1IzMVMzUjBTM1IzUVMzUzFTM1MxUzNTM1IzUjFSM1IxUjNSMVIxElHwczJwURFR8NMyEzPw01ESsBLwg9ASEPDgJ9Pj7+xz8/Xry82z4+/sc/Pz8fvB8/Hx8/H7wfPx8BWAEBawQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgsKCACFAQH+qAoJCQkICAcHBgUFBAMCAGElH15eXj+cPl5eXj4fHz4+Hx/+qCagPz8gIAFY+gYGBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAGICAwQFBQYHBwgICQkJCgJSAQEFBwgKCwYHBvOBAQIDBAUFBwYHCAGJCQkAAAAABwAAAAADdwP0AAMABWAjAEcAaAcPAlQAACUzNSM3FSM1NxUjFTMVIXUzFSMVMxUjNSM1MzUjNTM1IzUzNSCRHwYzITM/BhEvByEPBiUFbXEPBiEvBhE/BgMRFR8NMyEzPw01ESsBLwg9ASEjDw0FHwczJwHhPj5dfX0+Pj4+Pj4+Pz8/Pz8/2wEBAGIDAwQEAY4EBAMDAGIBAQEBAGIDAwQE/nIEBAMDAGIBAAEFcGkIBwYEAGIEBgcICQr+aAoJCAcGBAICBAYHCAkKqWICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgsKCACFAQH+qAoJCQkICAcHBgUFBAMCAGH0AQEDBAUFBgbb+ucfH11d+h8fIB8fHyAgHx8gHx8fDP5yBAQDAwMBAgIBAwMDBAQBJgQDBAMCAGEBAGECAGMEAY8BAGMGBwgJCv5oCgkJBwUEAGIEBQcJCQoBmAoJCAcGBAIBOPzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAGJCQkKAlIBAUHCAoLBgcG+gICAwQFBQYHBwgICQkJhwYGBQUEAGIB+gAAAAAHAAAAAN3A/QAAgAVADMAOWBDAE4AjwAAATMnFzsBPwU9AS8FKwE3HwwPBxcVIycjFSM1IxcjJyMHIzCjFSMVIzUjNQEfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80Adw2G4whCacGBQQDAgICBAUGBwgiIQgODQUFBQQAeAwICAgEBAQMEBQYHCCsjJiQhXEcID0kPIkc/OyA6AWgBAQMEBQUGBtv6/gwCAGMEBQUGBwICAKJCQoCMgoJCQkICAcHBgUFBAMCAvoGBwYLCggHBQEB/qgKCQkJCAGHBwYFBQQDAgIBa04xAgIEBAYGBwGBQUdAwIbAQEEAwMDBAQFBQUGDQkICAcGBQQETgJISLy8LCy8G6GhGwE4BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAGJCQkKAlIBAUHCAoLBgcG+gEBAGMEBQUHBgcICAKJCQAAAAgAAAAA3cD9AAVADgARABfALYA5gdXcATIAABMzPwg1LwgjFw8EHwc/CC8IDwIlFzcZBxcjJwcjNychHwsPCyM1BR8GIy8HIw8GFR8HMz8GMw8NIy8KNT8KMx8EJx8GHQEPCSSBLwk9AT8JOWEfARMfBzMnBREVHw0zITM/DTURKwEvCD0BIQ805RUMCwUEBACfAwIBBAUGBAQFBQsdsGIFAwEBAGMEBggECQoKCQQDBwUEAGEBAGQFBwMECQsKCAgBTicoJTg6JikoJJo5/jYMCwsKCQgHBgQDAgEBAQQEAgICGoLCw02AdwEBAQDAgIDIQEIDAwUFBggIBgkIBwYEAwIBAgMFBggICgkIBgYEBACIQMCAGMEBAUFBQYGBwPCwoKCQgHBgUEAwIBAQIDBAUGBwkJCgoLDw4GBQa6CAGGBQUdAgIDBAYGBwkJCgoLCwsKCQgIBgUFAwICAwQGBggICQoKCwwKCQUBAQMEBQUGBtv6/gwCAGMEBQUGBwICAKJCQoCMgoJCQkICAcHBgUFBAMCAvoGBwYLCggHBQEB/qgKCQkJCAGHBwYFBQQDAgIBPwEDAGMECAkMDRkOCwoIAwMCAGIOBQoLDhONDAoJBgIEAQEDAwMHCQsNDxgNCwoHAWIDAQEDBSJERF1fRUvfXQECBAQHbwgKCgwMDRYMCwsJCQcGBQMCAbwPBAUFBgYHDgkIBwUEAwEBAGQGCakMDRgODAsJBwUDAQIDBAUGCAkOBgYGBQUFBAMDAGIBAQIEBAYHCAkLCwsNEQ0MDAOKCAcGBQQAQMCAGQCBgcJCQsMDA4XDQwKCgkHBgUEAGIEBQYHCQoKDAwNGAwMCwoICAYFAwMDAwE8BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAGJCQkKAlIBAUHCAoLBgcG+gEBAGMEBQUHBgcICAKJCQAGAAAAAN3A/QABAAkAEcAbAB3ALgAAAEjNxc/ARUPBi8GPQE/Bh8GJREfByE/BxEvBiMhIw8EJRczHwYRDwchLwCRPwclHwczJwURFR8NMyEzPw01ESsBLwg9ASEPDgJ9+ipTU0kCAwQFBQYGBgYGBAQDAgIDBAQGBgYGBgUFBAMC/qgBAQECAwIDBAF1BAMDAGIBAQEBAGECAGMDBP6LBAMCBQEBAYUFBQkIBwYCBAlBAwQGBwQICv6GCgkIBwYCBAlBAwQGBwQJCQEtAQEDBAUFBgbb+v4MAGIDBAUFBgCHCAGJCQkKAjIKCQkJCAGHBwYFBQQDAgL6BgGCwoIBwUBAf6oCgkJCQgIBwcGBQUEAwICAQZ1TnU/BwUGBAQDAgEBAGMEBAYFBwYGBQUEAwEBAQEDBAUFBkf+qgQDAGMCAQEBAQEBAGMCAwQBVgMEAGMCAQEBAQUdAY0BAwQGBwQJCf6lCgkIBwYCBAlBAwQGBwQJCQFbCgkIBwYCBAG8BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAGJCQkKAlIBAUHCAoLBgcG+gEBAGMEBQUHBgcICAKJCQAHAAAAAN3A/QAAwAHAAsADwAZAD4AfWAAATM1IzsBNSMhMzUjOwE1IycVMzUzFTM1MxUjFTMVIXUzFSM1IxUjNSMVIzUzNSM1MzUjNSUfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80AeE+Pj4/P30/Pz8+Pj8/Pj8+Pj4+Pj4/Pj8+Pj4+PgEZAQEDBAUFBgbb+v4MAGIDBAUFBgCHCAGJCQkKAjIKCQkJCAGHBwYFBQQDAgL6BgGCwoIBwUBAf6oCgkJCQgIBwcGBQUAwICASU/Pj4+Pz4+Pj4+Pj8+Pz4+Pj4+Pj8+Pz76BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAGJCQkKAlIBAUHCAoLBgcG+gEBAGMEBQUHBgcICAKJCQAAAA0AAAAA3cD9AARACMANgBJAFMAVwCnALkAywDdAPAA+wE8AAALDwc1PwcHFS8HNx8GJQ8IJz8HIR8IBY8HJRujFTMVIxUjNSMVIzUHHwQjLwYrAQ8GFR8HMz8CNSM1MxUPBy8LNT8OHwIFIY8HNx8GJQ8HIz8HJQcvBzU

fBicjDwcnPwgfBzMnBREVHw0zITM/DTURKwEvCD0BIQ8OAqYQEBITFBQVFRISEhARDw8NoBUVFBQ  
TEhAQFA4PDxAREhIBGwEBBAYHCQoMDRULCgkIBgUEAv5GAQEEBQYICQoLFQ0MCgkHBgQCAaJOREQ  
dJB01AwQGBAICAgIDBAUGBgCFCQcHBQQCAGECBAUGBwkKCgkHBSI/BQUHCAgJCQoLCQkJBwcGBQQ  
DAQEBAGICAwQEBQUGBgCHBwgnDAsBPB4CAwUGCAkKCxUNDAoJBwYE/mILCgkIBgUEAh0CBAYHCQo  
MDQFhFQ0PDxAREhISFRUUFMBSEKUJCRISERAPDw0VEBASExQUFRWMAQEDBAUFBgbb+v4MAgIDBAU  
FBgCHCAgJCQkKAjIKCQkJCAGHBWYFBQQDAGL6BgCGCwoIBWUBAf6oCgkJCQgIBwcGBQUEAwICyA0  
MCgkHBgQCHgIDBQYICQoLNx0CBAYHCQoMDRUMCGgIBgUEzAsKFRQUExIQEBUNDw8QERISEgkJEH  
REA8PDRUQEBITFBQVFU8XMhdHp6e0DAQECQsMBWYFBQMCAGIDBQgICwWTDQsKCAcFAgECawUhFkE  
FBQQEAgIBAQECawQFBwcICgoKDBYJCAGHBWYFBQUDAwMBAQEBAwQsEhISEBEPDw0VEBERExQUFW0  
NDw8REBISEhUVFBQTEREQFRULCgkIBgUDAx0CBAYHCQoMFAIDBQYICQoLFQ0MCgkHBgQcNAYGBQU  
EAWEB+1781AoJCQkICACHBgUFBAMCAGICAwQFBQYHBWgICQkJCgJSAQEFBwGKCWYHBvoBAQIDBAU  
FBWYHCAGJCQkAAAAABWAAAAADdwP0ABUALAA3AE0AuQDEAQUAAAEPBhUfBj8GHwQ7AT8FNS8GBYc  
PAT8DLwQxDRQdAR8CPwI1LwUfAhUPAh8DPwIfCh0BDwsjLwUPDC8JPw4vBT8JMx8GNx8HMYcFERU  
fDTMhMz8NNRErAS8IPQEHdW4BgQsVDgYEAWECAQMDAwQDBAQDBQca5AkUCAKEBAUFBBQCAgEBAwM  
EBhEKEZsNDxoZGRgQEBAPHAUFBAIBAw8OCAMBAQICAgCHMgIBAQISFRYXGBYVfGSLDQoIBWUDAwE  
BAQIDAwMEBQUFBgUGBgkICAcNMDIfICERHAcFBQYHCAgIBwcHBgUFAwMBAQIFBwGJCQlGCwoKCAg  
HFhUEAgMBAQMFawQFBQYFBwoICAgHBWUEEQUEBAWQFBQYG2/r+DAICAwQFBQYHBWgICQkJCgIyCgk  
JCQgIBwcGBQUEAwIC+gYHBGsKCAcFAQH+qAoJCQkICACHBgUFBAMCAGelBAGIBAQFBQMFawMDAGe  
BAQEBAaOuPaCTBwUCAGMEBQQFBQMEAGICAgEBPiQiCAYGBRESEhOZAgIEBAUFBQkZFR8MDAYFBAM  
DBAIGCQgJCQhDgXoZGAQDAQEBAgQFBQYFBgYGBgUGBgUFBQUEBAMCAGEBAGMEciwJCAkLHzELBQQ  
EAWIBAQEDBAQGBgCkCgkJCQcHBgUEGxcXGBgXGSEhCgUKCwoKCQUEAwQDAGEBAGMFBQYHVQYGBQU  
EAWEB+1781AoJCQkICACHBgUFBAMCAGICAwQFBQYHBWgICQkJCgJSAQEFBwGKCWYHBvoBAQIDBAU  
FBWYHCAGJCQkABQAAAAADdwP0AAcAEwAbACYAZwAAARUjFSM1IzUjFzczBxcjJwcjNycjFSMVIzU  
jNQEfBzMnBREVHw0zITM/DhErAS8JNSEjDw0C8DogOoEoJyY50iYoKSY60Aw7IDoBaAEBAWQFBQY  
G2/r+DAICAwQFBQYHBWgICQkJCgIyCgkJCQgIBWYHBQUEAwIBAfoGBWYLCgGHBQEBAf6pCgkJCQg  
IBwcGBQUEAwICAeEboaEbRERdX0VFX10boaEbATgBgUFBAMBAfpe/NQKCQkJCAGHBWYFBQQDAGI  
CAGMEBQUGBwcICAKJCQoCUgEBBQcICGsGBWb6AgIDBAUFBgCHCAgJCQkAAAAABWAAAAADdwP0ABI  
AHgA+AGcAuWDGAQcAAAEzPwY9AS8FKwElFSMVMxUjFTMVIzUjFzMfCxUPCyMVIzUjFQ8KLWkzFR8  
FOWE/BT0BBR8FIY8GKwEPBh0BHwczPwI1IzUzFQ8GKwEvCjU/DTsBHwMDHwczJwURFR8NMYezPw0  
1ESsBLWg9ASEjDw0BdycJBWYFBAMCAGMEBQYHCCgBa1lNTVp7YQgHDgYFBgQFAwQAGIBAQICAgM  
EBAUFBg0OLyAlAQICAgMDBAKLCw4NDQoFCAMDAGMBIQUEDAwQFBgYHBQUEBAICaf0EBAQDBAMgAgI  
EBAYGBWgGCQkHBQUCAgMEBQcEBQkLDakIBYdHBgYICAKKCwsLCwoKCAgGBgQDAGEBAGIDAwUEBgY  
GBwcICAKIDgWGBYwBAQMEBQUGBtV6/gwCAGMEBQUGBwcICAKJCQoCMgoJCQkICACHBgUFBAMCAvo  
GBWYLCgGHBQEBAf6pCgkJCQgIBWYHBQUEAwIBAfoGBWYLCgGHBQEBAf6pCgkJCQgIBWYLCgGHBQEBAf6pCgkJCQg  
EBAUGBQYGDgYGBgUEBQQDBAIEAKa8hAYMBgUFBAGBQMBAGIFAwcFBAULDQcGBgQDAGICAwMFBgY  
HhA0EBQUFDA4IBWYFAwMCAGQGCAoLDhUPDAsJCAMCAwEDAwUlGEcHBQUEAwIBAgQEBgCJCQoLDA0  
ZCgkJCAcHBgYFBAMDAgECBAIEAUIGBgUFBAMBAfpe/NQKCQkJCAGHBWYFBQQDAGICAgMEBQUGBwc  
ICAKJCQoCUgEBBQcICGsGBWb6AgIDBAUFBgCHCAgJCQkAAAUAAAAA3cD9AAAFABEAggCNAM4AAAE  
VMxUjNSMXNzMHfYmNByM3JwUfBhUjLwYrAQ8FHQEfDRUPCiMvCjMVHwU7AT8GLw89AT8JMx8BAx8  
HMYcFERUfDTMhMz8NNRErAS8IPQEHdW4B61V2jignJk6JigpJjs5AaMHBWYEBAMCIAECAGUFBgc  
IBwcGBQQCAGIDBA0mCAGGBQUDAwIBAQECawMEBAoLDQ8JCgkJCAcGBQUDAQEHAgQEBgCICQgHBgU  
EAgIBAQICBAMJKAgiBgUEBAMCAQEBAgMDCAoLDQ4KCQg0AQEDBAUFBgbb+v4MAgIDBAUFBgCHCAg  
JCQkKAjIKCQkJCAGHBWYFBQQDAGL6BgCGCwoIBWUBAf6oCgkJCQgIBwcGBQUEAwICAeGiGrxERf1  
fRUVfXQUEBQUGBwGHCAGBQUEAgIBAwIEBQUFBQUEBAcNBAQEBAUFBgYGBwsFBQUEBAMHBAIBAgI  
DBAUGBgCHCAkIBgYFBAICAQIDBAQFBgUGBAQBBQ4EBAUEBQUBgYGBgUFBQUECAYFAgECAGe6BgY  
FBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAgJCQkKAlIBAUHCAoLBgcG+gEBAGM  
EBQUHBgCICAKJCQAAAAQAAAAA3cD9AADAGIAbQCuAAABFTc1Nx8CFREVDWkjLwk9AT8KOWEXNQc  
VDwkrAS8IPQE/CjsBFz0BPwU7ARcnHwczJwURFR8NMYezPw01ESsBLWg9ASEPDGvxx8DAGICAwQ  
ICQsMCAkICQkJCAcHBgQEAgECAQMDCAkLDAGJCAkICcMCAwQICQsMCAkICQkJCAcHBgQEAgECAgI  
EBWkLDAGJCAkJCAEEBQMD6wQDBxUBAQMEBQUGBtV6/gwCAGMEBQUGBwcICAKJCQoCMgoJCQkICAc  
HBgUFBAMCAvoGBWYLCgGHBQEBAf6pCgkJCQgIBWYHBQUEAwIBAfoGBWYLCgGHBQEBAf6pCgkJCQgIBWYLCgGHBQEBAf6pCgkJCQg  
CAGEBAwQFBgYIBQcGBgYHBgYKQCGAwMBAp0n1QkJCQoJCAYDAGICAwQFBQcHBgYHBgYGBWYKQCG  
GAwICAvGEAwYFAgEvAt0GBgUFBAMBAfpe/NQKCQkJCAGHBWYFBQQDAGICAgMEBQUGBwcICAKJCQo  
CUgEBBQcICGsGBWb6AQECawQFBQcGBWgICQkJAAAAAASAN4AAQAAAAAABAAAAAQAQAAAAAQA  
NAAEAQAQAAAAAQAHA4AAQAAAAAAwANABUAAQAAAAAABAANACIAAQAAAAAABQALAC8AAQAAAAA  
ABgANADoAAQAAAAAACgAsAEcAAQAAAAAACwASAHMAAwABBAkAAAACAIUAwABBAkAAQAaAICaAwA  
BBakAAgAOAKEAAwABBAkAAwAaAK8AAwABBAkABAAaAMkaAwABBAkABQAWAOMAawABBAkABgAaAPk

```
AAwABBAkACgBYARMAAwABBAkACwAkAWsgRmlsZVR5cGVfRm9udFJlZ3VsYXJGaWxlVHlwZV9Gb250RmlsZVR5cGVfRm9udFJlcnNpb24gMS4wRmlsZVR5cGVfRm9udEZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAARgBpAGwAZQBUAHkAcABlAF8ARgBvAG4AdABSAGUAZwB1AGwAYQByAEYAaQBsAGUAVAB5AHAAZQBfAEYAAbwBuAHQARgBpAGwAZQBUAHkAcABlAF8ARgBvAG4AdABWAGUAcgBzAGkAbwBuACAAMQAuADAARgBpAGwAZQBUAHkAcABlAF8ARgBvAG4AdABGAG8AbgB0ACAAZwB1AG4AZQByAGEAdABlAGQAIAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAaGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAAQ4BDwEQAREBEgETAANCTVAEWExTWANBVkkDRE9DA0VYRQNNUDQDwklQA1RBUGRET0NYA0pQRwNQTKcDR0lGA1BERgNUWFQESlBFRwNYTFMDTVazAAA=) format('truetype');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'FileType_Font' !important;
speak: none;
font-size: 35px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
position: relative;
top: 8px;
}
/* csslint ignore:end */
```

## Custom template

You can design the own template by preventing the default file list including buttons. The [showFileList](#) property used to display whether the default file list or own file list When you use custom template to upload or remove the files, pass the custom UI argument as true to call `upload/remove` public method as follows:

- `UploaderObj.upload(filesData, true);`
- `UploaderObj.remove(filesData, true);`

Refer to the following code sample.

## INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
import { Uploader, FileInfo } from '@syncfusion/ej2-inputs';
import { createElement, isNullOrUndefined, detach, EventHandler } from '@syncfusion/ej2-base';
enableRipple(true);
let dropElement: HTMLElement =
document.querySelector('.control_wrapper') as HTMLElement; let filesDetails
: FileInfo[] = [];
let fileList: HTMLElement[] = [];
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
```

```

 saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 }, dropArea: dropElement, selected: onFileSelect, progress:
onFileUpload, success: onUploadSuccess,
 failure: onUploadFailed, removing: onFileRemove,
 });
 uploadObj.appendTo('#fileupload');
 document.getElementById('browse').onclick = () => {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click(); return false;
 };
 document.getElementById('clearbtn').onclick = () => {
 uploadObj.element.value = '';
 detach(document.getElementById('dropArea').querySelector('.upload-
list-root')); filesDetails = []; filesList = [];
 };
 let parentElement : HTMLElement; let proxy : any; let
progressbarContainer : HTMLElement;
 function onFileSelect(args : any) : void {
 if
(isNullOrUndefined(document.getElementById('dropArea').querySelector('.uploa
d-list-root'))) {
 parentElement = createElement('div', { className: 'upload-list-
root' });
 parentElement.appendChild(createElement('ul', {className: 'ul-
element' }));
 document.getElementById('dropArea').appendChild(parentElement);
 }
 for (let i : number = 0; i < args.filesData.length; i++) {
formSelectedData(args.filesData[i], this); }
 filesDetails = filesDetails.concat(args.filesData);
 this.upload(args.filesData, true); args.cancel = true;
 }
 function formSelectedData (selectedFiles : FileInfo, proxy: any) :
void {
 let liEle : HTMLElement = createElement('li', { className: 'file-
lists', attrs: { 'data-file-name' : selectedFiles.name } });
 liEle.appendChild(createElement('span', {className: 'file-name ',
innerHTML: selectedFiles.name }));
 liEle.appendChild(createElement('span', {className: 'file-size ',
innerHTML: proxy.bytesToSize(selectedFiles.size) }));
 if (selectedFiles.statusCode === '1') {
 progressbarContainer = createElement('span', {className:
'progress-bar-container'});
 progressbarContainer.appendChild(createElement('progress',
{className: 'progress', attrs: {value : '0', max : '100'}}));
 liEle.appendChild(progressbarContainer);
 } else { liEle.querySelector('.file-name').classList.add('upload-
fails'); }
 let closeIconContainer : HTMLElement = createElement('span',
{className: 'e-icons close-icon-container'});
 EventHandler.add(closeIconContainer, 'click', removeFiles, proxy);
 liEle.appendChild(closeIconContainer); document.querySelector('.ul-
element').appendChild(liEle);
 filesList.push(liEle);

```

```

 }
 function onFileUpload(args : any) : void {
 let li : Element =
document.getElementById('dropArea').querySelector('[data-file-name="' +
args.file.name + '"');
 EventHandler.remove(li.querySelector('.close-icon-container'),
'click', removeFiles);
 let progressValue : number = Math.round((args.e.loaded /
args.e.total) * 100);
 if (!isNaN(progressValue)) {
li.getElementsByTagName('progress')[0].value = progressValue; }
 }
 function onUploadSuccess(args : any) : void {
 let li : Element =
document.getElementById('dropArea').querySelector('[data-file-name="' +
args.file.name + '"');
 if (!isNullOrUndefined(li.querySelector('.progress-bar-container')))
{
 detach(li.querySelector('.progress-bar-container')); }
 if (args.operation === 'upload') {
 li.querySelector('.file-name').classList.add('upload-success');
 li.querySelector('.close-icon-container').classList.add('delete-
icon');
 }
 if (args.operation === 'remove') {
 filesList.splice(this.fileList.indexOf(li), 1);
filesDetails.splice(this.fileList.indexOf(li), 1);
 }
 EventHandler.add(li.querySelector('.close-icon-container'), 'click',
removeFiles, this);
 }
 function onUploadFailed(args : any) : void {
 let li : Element =
document.getElementById('dropArea').querySelector('[data-file-name="' +
args.file.name + '"');
 EventHandler.add(li.querySelector('.close-icon-container'), 'click',
removeFiles, this);
 li.querySelector('.file-name').classList.add('upload-fails');
 if (args.operation === 'upload')
{detach(li.querySelector('.progress-bar-container')); }
 }
 function removeFiles(args : any) : void {
 if (!isNullOrUndefined(args.currentTarget)) {
 if
(filesDetails[filesList.indexOf(args.currentTarget.parentElement)].statusCod
e === '2') {

this.remove(filesDetails[filesList.indexOf(args.currentTarget.parentElement)
]);

filesDetails.splice(filesList.indexOf(args.currentTarget.parentElement), 1);
 } else { onFileRemove(args); }
 }
 }
 function onFileRemove(args: any) : void {
 if (!isNullOrUndefined(args.currentTarget)) {

```

```

 if
 (filesDetails[filesList.indexOf(args.currentTarget.parentElement)].statusCod
e !== '2') {
 detach(args.currentTarget.parentElement);
 filesList.splice(filesList.indexOf(args.currentTarget.parentElement), 1);
 }
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div id="dropArea">
 Drop files here or <a href=""
id="browse"><u>Browse</u>
 <input type="file" id="fileupload">
 </div>
 <div style="margin-left: 50px; padding-top:25px;">
 <button class="e-btn e-css" id="clearbtn">Clear All</button>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## STYLES.CSS



```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.control_wrapper {
 max-width: 500px;
 margin: auto;
}
.e-upload {
 width: 100%;
}
.e-file-select-wrap {
 display: none;
}
.e-upload {
 border: none;
 margin-top: 15px;
}
#drop {
 padding-left: 30%;
 font-size: 14px;
 font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif"
}
#dropArea {
 min-height: 18px;
 border: 1px dashed #c3c3cc;
 padding: 15px 0;
 margin: 0 auto;
 width: 440px;
}
.wrapper{
 padding: 0 10px;
}
.e-upload-success, .e-upload-failed {
 display: none;
}
.sf-icon-bmp:before { content: "\e700"; }
.sf-icon-xlsx:before, .sf-icon-XLSX:before { content: "\e701"; }
.sf-icon-avi:before, .sf-icon-AVI:before { content: "\e702"; }
.sf-icon-doc:before, .sf-icon-DOC:before { content: "\e703"; }
.sf-icon-exe:before { content: "\e704"; }
.sf-icon-mp4:before { content: "\e705"; }
.sf-icon-zip:before, .sf-icon-ZIP:before { content: "\e706"; }
.sf-icon-tar:before { content: "\e707"; }
.sf-icon-docx:before { content: "\e708"; }
```

```
.sf-icon-jpg:before { content: "\e709"; }
.sf-icon-png:before { content: "\e70a"; }
.sf-icon-gif:before { content: "\e70b"; }
.sf-icon-pdf:before { content: "\e70c"; }
.sf-icon-txt:before { content: "\e70d"; }
.sf-icon-jpeg:before { content: "\e70e"; }
.sf-icon-xls:before { content: "\e70f"; }
.sf-icon-mp3:before { content: "\e710"; }
#dropArea .e-upload .e-upload-files .e-file-delete-btn,
#dropArea .e-upload .e-upload-files .e-file-remove-btn {
 top: 35%;
}
span.file-icon{
 width: 25px;
 height: 25px;
 display: inline-flex;
 background-size: contain;
 margin: 7px;
}
img.file-icon {
 width: 20px;
 height: 20px;
}
.file-name {
 color: #e1e1e1;
 font-size: 14px;
 padding: 3px 10px;
 overflow: hidden;
 text-overflow: ellipsis;
 display: inline-block;
 width: 50%;
 white-space: nowrap;
 position: relative;
 top: 5px;
}
.file-size {
 font-size: 12px;
 padding: 3px 10px;
 overflow: hidden;
 display: inline-block;
 position: relative;
 top: 6px;
}
.progressbar{
 height: 5px;
 width: 70%;
 margin-left: 14px;
}
.upload-success{
 background-color: green;
}
.upload-failed{
 background-color: red;
}
#dropArea .e-upload .e-upload-files .e-upload-file-list {
 min-height: 63px;
}
```

```
#dropArea .e-upload {
border-width: 0 1px;
}
/* csslint ignore:start */
@font-face {
font-family: 'FileType_Font';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSgIAAAEoAAAVmNtYXDNyOfNAAABYAAAAFznBhlmBlU
kdGAAAKgAAB78aGVhZA/zy4MAAADQAAAAANmhoZWEH1AQTAARAAAAACRobXR4SAAAAAAAAAYAAAB
IbG9jYUgKP2YAAAIgAAAAJmlheHABKQFJAAABCAAAACBuYwllPdPGuQAAIUQAAAJtcG9zdGQHRBs
AAC00AAAAkAABAAAAEAAAAAFwEAAAAAADdwABAAAAAAAAAAAAAAAAAAAAAAEgABAAAAAQAAvWuGj18
PPPUACwQAAAAAAAAAiQ0kAAAAA1qJDSQAAAAADdwP0AAAAACAACAAAAAAAAAAAAEAAAAASAT0ADQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAazAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnEAQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAAAAAAAgAAAAAMAAAUAMAAQAAABQABABCAAAABAAEAAEAOCQ//8AAOCa//8AAAAABAAQ
AAAABAAIAAwAEAAUABgAHAAGACQAKAAsADAANAA4ADwAQABEAAAAAAAAABCGHaAswDrAQ4BNQFSaZ
iB8wIsAlKCsgMAAyEDbgOrg9+AAAAACAAAAAADdwP0ABIAJQA4AFgAZwCRANIA3QAAATM/Bj0BLwY
jJTM/Bj0BLwUrAQU7AT8GLwcjJRczHwsVDwsjFSM1Ixc3MxUjNtChIYcXFSM1IzMfChUPBh8HDwo
jNQMRFR8NMyEzPw01ESsBLwg9ASEjDw0FHwczJwEjJQCGBgUDawICAgMEBgYHJwGOJwGHBgUEAwI
CAwQFBGcHKP5yIAGGBQUEAwEBAQEDAwUFBwggAbUHCA0GBgUFBAQDAwIBAQEBAgMDBAQFBQYMDy4
hvTc2KiEDNxc4BCFmCAcNCwoDBAMCAgEBAGIDBAQGBGcGBgQEAWEBAGIBAwMDBAUkDA5MegICAwQ
FBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBGsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgH
0AQEDBAUFBgbb+gE/AQEDAwQFBgYHBgYHBgUEAwMBAQsBAQMDBQUHBwCBGUEAwI0AgIDBAQFBGcFBQM
DAgEBGgEDAwIEBAQEIBgYFBgYFBgUEAwMBAQsBAQRryQkLw+U5GRUz68AQIEBgMEBAUFBgYFBgY
FBQQEAWMDBQUGBGcIBwsFBQUEBAMFBAK8AbX81AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQk
JCgJSAQEgFBwgKCwYHBvoCAgMEBQUGBwCICakJCyCGBgUFBaICAfoAAAYAAAAAA3cd9AAZADMATQEB
ZAGQApQAAATMfAx0BDwMjISMvAz0BPwMzJTMfAx0BDwMrAi8DPQE/AzM3Mx8DHQEPaysClwM9AT8
DMycXNzMHfyMnByM3JwEfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80AuoDAwUEAQEEBQMD/iwEAWQ
EAQEEBAMEAdQDAwUEAQEEBQMD2gQDBAQBAQEEAwTaAwMFBAEBBAUDA9oEAWQEAQEEBAME1SssJ0B
CJy4tKENBAW0BAQMEBQUGBtv6/gwCAgMEBQUGBwCICakJCQoCMgoJCQkICAcHBgUFBAMCAvoGBwY
LCggHBQEB/qgKCQkICAgHBwYFBQQDAgIBBgEEBQMDAwMFAwEBawUDAwMDBQQBfQEEBQMDAwMFAwE
BAwUDAwMDBQQBfQEEBQMDAwMFAwEBawUDAwMDBQQBCUREXF9GR19cARAGBgUFBAMBafpe/NQKCQk
JCAgHBwYFBQQDAgICAgMEBQUGBwCICakJCQoCUgEBBQcICGsGBwB6AQECawQFBQcGBwgICQkJAAA
ABQAAAAADdwP0AAIAQwCEAI8A0AAAAQc1BxUfDz80PQEvdg80BQcVDw4vDz8PHw4DHwczJwURFR8
NMyEzPw01ESsBLwg9ASEPDgJefX0BAGQGBwKcCwNDw8PERESERERDw8PDQwLCgkHBgQDAwQGBwK
KCwNDw8PEREREHERDw8PDQwLCgkHBgQCAXYBAwUHCQoMDQ4QERETFBQUFRQUEhIREA4NDAoJBgU
DAgIDBQYJCgwNDhAREhIUFBQUFBQTEREQDg0MCgkHBQNDaQEDBAUFBgbb+v4MAgIDBAUFBgCHAg
JCQkKAjIKCQkICAgHBwYFBQQDAg16BgCGCoIBwUBAf6oCgkJCQgIBwcGBQUEAwICAXN0U8JCBE
REAS0DQwLCgkHBgQDAQEDBAYHCQoLDA0ODxARERESERASQDw40DAsKAgFBQIBAQIFBQgICFgSMdg4
PEBAREgoKFRMTEREQDg0MCgkHBQMBAQMFBwKDA0OEBERExMVFBUUFBISEQ8PDQwKCAcFawEBawU
HCAoMDQ8PERISFBQBkQYGBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJ
SAQEgFBwgKCwYHBvoBAQIDBAUFBWYHCAgJCQkAAAYAAAAAA3cd9AAZADMATQBMaHEAsgAAATMfAx0
BDwMjISMvAz0BPwMzJTMfAx0BDwMrAi8EPwQzNzMfAx0BDwMrAi8EPwQzJR8CMz8CMx8BMz8BMwc
jLwIxDwEjJyUfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80AwoDAwUDAQEDBQMD/gwEAWQEAQEEBAM
EafQDAwUDAQEDBQMDvAMDBQMBAQEBAwUDA7wDAwUDAQEDBQMDvAMDBQMBAQEBAwUDA/7AJwECAQE
DKXsnAwEBKR04HycCAQIShjkBjgEBawQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwc
GBQUEAwIC+gYHBGsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgElAQMFawMEAgUEAQEEBQIEAwMFAwF
9AQMFawMEAgUEAQEEBQIEAwMFAwF9AQMFawMEAgUEAQEEBQIEAwMFAwEBhgMNCAMfHhAHj7J/CAK
HibL5BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHAgJCQkKALIBAUHCAoLBgc
G+gEBAGMEBQUHBGcICakJCQAAAAFAAAAAAN3A/QACwAXACMALGbVAAABFSMVMxUjFTMVIzUjFzc
zBxcjJwcjNycjFSMVMxUjFTMVIzUBHwczJwURFR8NMyEzPw01ESsBLwg9ASEPDgLYWU1NWnuOKCc
mOTomKCKmOzkPWU1NWnsBWwEBAwQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQU
EAWIC+gYHBGsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgHhGhMaOhq8RERD10bMx06GrwBOAY
GBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJSAQEgFBwgKCwYHBvoBAQI
DBAUFBWYHCAgJCQkAAAAACAAAAAADdwP0AAQACAAMABEAFAQAxADwAfQAAARUzNSMFMzUjFzMIzZM
```

VMzUjBTM1IzUVMzUzFTM1MxUzNTMRIzUjFSM1IxUjNSMVIxElHwcZJwURFR8NMMyEzPw01ESsBLwg9ASEPDgJ9Pj7+xz8/Xry82z4+/sc/Pz8fvB8/Hx8/H7wfPx8BWAEBawQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgElH15eXj+cPl5eXj4fHz4+Hx/+qCagPz8gIAFY+gYGBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJSAQEFBwgKCWYHBvoBAQIDBAUFbWYHCAgJCQkAAAAABwAAAAADdwP0AAMABwAjaEcaAaCpALQAACUzNSM3FSM1NxUjFTMVIxUzFSMVMxUjNSM1MzUjNTM1IzUzNScRHwYzITM/BhEvByEPBiUfBxEPBiEvBhE/BgMRFR8NMMyEzPw01ESsBLwg9ASEjDw0FHwcZJwHhPj5dfX0+Pj4+Pj4+Pz8/Pz8/2wEBagIDAwQEAY4EBAMDagIBAQEBagIDAwQE/nIEBAMDagIBAAEFcGkIBwYEagIEBgCICQr+aAoJCACGBAICBAYHCAkKqWICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgH0AQEDBAUFbGbb+ucfH11d+h8fIB8fHyAgHx8gHx8fDP5yBAQDAwMBAgIBAwMDBAQBJgQDBAMCAgEBAQECAgMEAY8BAgMGBwgJCv5oCgkJBwUEagIEBQcJCQoBmAoJCACGBAIBOPzUCgkJCQgIBwcGBQUEAwICAgIDBAUFbGcHCAgJCQkKALIBAQUHCAoLBgcG+gICAwQFBQYHBwgICQkJhWYGBQUEAgIB+gAAAAHAHAHAAN3A/QAAgAVADMAOWBDAE4AjwAAATMnFzsBPwU9AS8FKWE3HwwPBxcVIycjFSM1IxcjJyMHIZcjFSMVIzUjNQEfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80Adw2G4whCacGBQQDAgICBAUGBwgiIQgODQUBQQAeAwICAgEBAQMEBQYHCCsjJiQhXECid0kPIkc/OyA6AWgBAQMEBQUGBtv6/gwCagMEBQUGBwCICakJCQoCMgoJCQkICAcHBgUFBAMCAvoGBwYLCggHBQEB/qgKCQkJCAGHBwYFBQQDAgIBA04xAgIEBAYGBwCGBQUDAwIbAQEEAwMDBAQFBQUGDQkICAcGBQQETgJISLy8LCy8G6GhGwE4BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgcHCAgJCQkKALIBAQUHCAoLBgcG+gEBAGMEBQUHBgcICakJCQAAAAgAAAAA3cD9AAVADgARABfALYA5gDxATIAABMzPwg1LwgjFw8EHwc/CC8IDwIlFzczBxcjJwcjNychHwsPCyM1BR8GIy8HIw8GFR8HMz8GMw8NIy8KNT8KMx8EJx8GHQEPcSsBLwk9AT8JOWEfARMfBzMnBREVHw0zITM/DTURKwEvCD0BIQ805RUMCWUEBACFAwIBBAUGBAQFBQsdsgIFAwEBAGMEBgGECQoKCQQDBwUEAgEBAGQFBwMECQsKCAgBTicoJTg6JikoJjo5/jYMCwsKCQgHBGQDAgEBAQQEBgcICgoLCw02AdwEBAQDAgIDIQEDAwUFBggIBgkIBwYEAwIBAGMFBggICgkIBgYEBaICIQMCAGMEBAUFbQYGBwCPCwoKCQgHBgUEAwIBAQIDBAUGBwkJCgoLDw4GBQa6CagGBQUDAgIDBAYGBwkJCgoLCwsKCQgIBgUFAwICAwQGBggICQoKCwwKCqUBAQMEBQUGBtv6/gwCagMEBQUGBwCICakJCQoCMgoJCQkICAcHBgUFBAMCAvoGBwYLCggHBQEB/qgKCQkJCAGHBwYFBQQDAgIBPwEDAgMECAkMDRkOCwoIAwMCAGIOBQoLDhONDAoJBgIEAQEDAwMHCQsNDxgNCwoHawIDAQEDBSJERF1fRUvfXQECBAQHbwKCGwMDRYMCwsJCQcGBQMCAbwPBAUFBgYHDgkIBwUEAwEBAGQGCakMDRgODAsJBwUDAQIDBAUGCAkOBgYGBQUFBAMDagIBAQIEBAYHCAKLCwsNEQ0MDAoKCAcGBQQAQMCawQCBgcJCQsMDA4XDQwKCgkHBgUEAgIEBQYHCQoKDAwNGAwMCwoICAYFAwMDAwE8BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFbGcHCAgJCQkKALIBAQUHCAoLBgcG+gEBAGMEBQUHBgcICakJCQAGAAAAAN3A/QABAakAEcAbAB3ALgAAAEjNxc/ARUPBi8GPQE/Bh8GJREfByE/BxEvBiMhIw8EJRczHwYRDwchLwCRPwclHwcZJwURFR8NMMyEzPw01ESsBLwg9ASEPDgJ9+ipTU0kCAwQFBQYGBgYGBAQDAgIDBAQGBgYGBgUFBAMC/qgBAQECAwIDBAF1BAMDagIBAQEBAGECAGMDBP6LBAMCBQEBAYUFbQkIBwYCBaIBAwQGBwQICv6GCgkIBwYCBaIBAwQGBwQJCQEtAQEDBAUFbGbb+v4MAgIDBAUFbGcHCAgJCQkKajIKCQkJCAGHBwYFBQQDAgL6BgCGCwoIBwUBAf6oCgkJCQgIBwcGBQUEAwICAQZ1TnU/BwUGBAQDAgEBAGMEBAYFBwYGBQUEAwEBAGEDBAUFbKf+qgQDAgMCAQEBAQEBAgMCAwQBVgMEAgMCAQEBAQUdAY0BAwQGBwQJCf6lCgkIBwYCBaIBAwQGBwQJCQFbCgkIBwYCBAG8BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFbGcHCAgJCQkKALIBAQUHCAoLBgcG+gEBAGMEBQUHBgcICakJCQAHAAAAAN3A/QAAwAHAAsADwAzAD4AfWAAATM1IzsBNSMhMzUjOwE1IycVMzUzFTM1MxUjFTMVIxUzFSM1IxUjNSMVIzUzNSM1MzUjNSUfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80AeE+Pj4/P30/Pz8+Pj8/Pj8+Pj4+Pj4/Pj8+Pj4+PgEZAQEDBAUFbGbb+v4MAgIDBAUFbGcHCAgJCQkKajIKCQkJCAGHBwYFBQQDAgL6BgCGCwoIBwUBAf6oCgkJCQgIBwcGBQUeAwICASU/Pj4+Pz4+Pj4+Pj8+Pz4+Pj4+Pj8+Pz76BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFbGcHCAgJCQkKALIBAQUHCAoLBgcG+gEBAGMEBQUHBgcICakJCQAAAA0AAAAA3cD9AARACMANgBJAFMAVwCnAlkAywDdAPAA+wE8AAALDwc1PwcHFS8HNx8GJQ8IJz8HIR8IBY8HJRujFTMVIxUjNSMVIzUHHwQjLwYrAQ8GFR8HMz8CNSM1MxUPBy8LNT8OHwIFIY8HNx8GJQ8HIz8HJQcvBzUfBicQJDFwcnPwgfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80AqYQEBITFBQVFRISEhARDw8NoBUVFBQTEHAQDA4PDxAREhIBGWEbbAYHCQoMDRULCgkIBgUEAw5GAQEEBQYICQoLFQ0MCgkHBgQCAaJOREQdJB01AwQGBAICAgIDBAUGBgcFCQcHBQCAgECBAUwBwKCGkHBSI/BQUHCAgJCQoLCQkIBwcGBQQDAQEBAgICAwQEBQUGBgcHBwgNDAsBPB4CAwUGCAkKCxUNDaOJBwYE/mILCgkIBgUEAh0CBAYHCQoMDQFhFQ0PDxAREhISFRUUFbMSEKUJCRISERAPDw0VEBASExQUFRWMAQEDBAUFbGbb+v4MAgIDBAUFBgcHCAgJCQkKajIKCQkJCAGHBwYFBQQDAgL6BgCGCwoIBwUBAf6oCgkJCQgIBwcGBQUEAwICyA0MCgkHBgQCHgIDBQYICQoLNx0CBAYHCQoMDRUMCGgIBgUEzAsKFRQUEXIQEBUNDw8QERISEgkJEHIREA8PDRUQEBITFBQVFU8XMhdHp6eoDAQECQsMBwYFBQMCAGIDBQgICwwTDQsKCAcFAgECawUhFkEfbQQEAgIBAQECawQFBwcICgoKDBYJCAgHBwYFBQUDAwMBAQEBAwQsEhISEBEPDw0VEBERExQUFW0NDw8REBISEhUVFBQTEREQFRULCgkIBgUDAx0CBAYHCQoMFAIDBQYICQoLFQ0MCgkHBgQCNAyGBQU

EAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJSAQEFBwgKCwYHBvoBAQIDBAU  
 FBwYHCAGJCQkAAAAABwAAAAADdwP0ABUALAA3AE0AuQDEAQUAAAEPBhUfBj8GHwQ7AT8FNS8GByc  
 PAT8DLwQxDwQdAR8CPwI1LwUfAhUPAh8DPwIfCh0BDwsjLwUPDC8JPw4vBT8JmX8GNx8HMyCfERU  
 fDTMhMz8NNRErAS8IPQEHdW4BgQsVDgYEAwECAQMDAwQDBAQDBQca5AkUCAkEBAUFBBQCAgEBAwM  
 EBhEKEZsNDxozGGRgQEBAPHAUFBAIBAw8OCAMBAQICAgcHMgIBAQISFRYXGBYVfGsLDQoIBwUDAwE  
 BAQIDAwMEBQUFBgUGBgkICAcNMdIfICERHAcFBQYHCAGIBwCHBgUFAwMBAQIFBwgJCQlGCwoKCAg  
 HFhUEAgMBAQMFawQFBQYFBwoICAgHBwUEEgEBAwQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgk  
 JCQgIBwcGBQUEAwIC+gYHBGsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgElBAGIBAQFBQMFawMDAgE  
 BAQEBBAouPacTBwUCAGMEBQQFBQMEAgICAgEBPiQiCAYGBRESEhOZAgIEBAUFBBQkZFR8MDAYFBAM  
 DBAIGCQgJCQhDGxozGAQDAQEBAGQFBQYFBgYGBgUGBgUFBQUEBAMCAgEBAgMECiwJCAkLHzELBQQ  
 EAwIBAQEDBAQGBgCkCgkJCQcHBgUEGxcXGBgXGSEhCgUKCwoKCQUEAwQDAgEBAgMFBQYHvQYGBQU  
 EAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJSAQEFBwgKCwYHBvoBAQIDBAU  
 FBwYHCAGJCQkABQAAAAADdwP0AAcAEwAbACyAZwAAARUjFSM1IzUjFzczBxcjJwcjNycjFSMVizU  
 jNQEfBzMnBREVHw0zITM/DhErAS8JNSEjDw0C8DogOoEoJyY5OiYoKSY60Aw7IDoBaAEBAwQFBQY  
 G2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwYHBQUEAwIBAfoGBwYLCggHBQEBAf6pCgkJCQg  
 IBwcGBQUEAwICAeEboaEbRERdX0VFX10boaEbATgBgUFBAMBafpe/NQKQKJCAGHBwYFBQQDAgI  
 CAgMEBQUGBwCICakJCQoCUgEBBQcICGsGBwb6AgIDBAUFBgHCAGJCQkAAAAABwAAAAADdwP0ABI  
 AHgA+AGcAuWDGAQcAAAEzPwY9AS8FKwElFSMVMxUjFTMVizUjFzMfCxUPCyMVIzUjFQ8KLwkzFR8  
 FOWE/BT0BBR8FIy8GKwEPbh0BHwczPwI1IzUzFQ8GKwEvCjU/DTSBHwMDHwczJwURFR8NMyeZPw0  
 1ESsBLwg9ASEjDw0BdycJBwYFBAMCAgMEBQYHCCgBa1lNTVp7YQgHDgYFBgQFAwQCAgIBAQICAgM  
 EBAUFBg0OLyAlAQICAgMDBAkLCw4NDQoFCAMDAgMBIQEDAwQFBgYHBQUEBAICaf0EBAQDBAMgAgI  
 EBAYGBwgGCQkHBQUCAGMEBQcEBQkLDakIBYdHBgYICakKCwsLCwoKCAgGBgQDAgEBAgIDAUEBgY  
 GBwCICakIDgWGBYwBAQMEBQUGBtv6/gwCAGMEBQUGBwCICakJCQoCMgoJCQkICAcHBgUFBAMCAvo  
 GBwYLCggHBQEBA/qgKCQkJCAGHBwYFBQQDAgIBhQEBAwMFBQcHBwYGBQQDAhshbMxo6GrwBAwMDAwQ  
 EBAUGBQYGDgYGBgUEBQQDBAIEAka8hAYMBgUFBAGUBQMBAGIFAwcFBAULDQcGBgQDAgICAwMFBgY  
 HhA0EBQUFDA4IBwYFAwMCAgQGCAoLDhUPDAsJCAMCAwEDAwULGEcHBQUEAwIBAgQEbgJCQoLDA0  
 ZCgkJCACHBgYFBAMDAgECBAIEAUIGBgUFBAMBafpe/NQKQKJCAGHBwYFBQQDAgICAgMEBQUGBwC  
 ICakJCQoCUgEBBQcICGsGBwb6AgIDBAUFBgHCAGJCQkAAAUAAAAAA3cD9AAAFABEAggCNAM4AAAE  
 VMxUjNSMXNzMHfYmNByM3JwUfBhUjLwYrAQ8FHQEfDRUPCiMvCjMVHwU7AT8GLw89AT8JMx8BAX8  
 HMyCfERUfDTMhMz8NNRErAS8IPQEHdW4B6lV2jignJjk6JigpJjs5AaMHBwYEBAMCIAECAGUFBgc  
 IBwcGBQQAICAgIDBA0mCAGGBQUdAwIBAQEACAwMEBAoLDQ8JCgkJCACGBQUdAwIEAgQEbgICQgHBgU  
 EAgIBAQICBAMJKAgiBgUEBAMCAQEBAgMDCAoLDQ4KCQg0AQEDBAUFBgbb+v4MAgIDBAUFBgHCAG  
 JCQkKajIKCQkJCAGHBwYFBQQDAgL6BgCGCwoIBwUBAf6oCgkJCQgIBwcGBQUEAwICAeGiGrxERf1  
 fRUVfXQUEBQUGBwgHCAGBQUEAgIBAwIEBQUFBQUEBAcNBAQEBOFBgYGBwsFBQUEBAMHBAIBAgI  
 DBAUGBgHCAGIBgYFBAICAQIDBAQFBgUGBAQBBQ4EBAUEBQUFBgYGBgUFBQUECAYFAgECAGe6BgY  
 FBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgHCAGJCQkKAlIBAUHCAoLBgcG+gEBAgM  
 EBQUHBgcICakJCQAAAAQAAAAAA3cD9AADAGIAbQCuAAABFTc1Nx8CFREVDwkjLwk9AT8KOWEXNQc  
 VDwkrAS8IPQE/CjsBFz0BPwU7ARcnHwczJwURFR8NMyeZPw01ESsBLwg9ASEPDgGvxx8DAGICAwQ  
 ICQsMCAkICQkJCAcHBgQEAgECAQMDCakLDAgJCAkICcMCAwQICQsMCAkICQkJCAcHBgQEAgECAgI  
 EBwkLDAgJCAkJCAEEBQMD6wQDBxUBAQMEBQUGBtv6/gwCAGMEBQUGBwCICakJCQoCMgoJCQkICAc  
 HBgUFBAMCAvoGBwYLCggHBQEBA/qgKCQkJCAGHBwYFBQQDAgIB7CQoJCYDBAUe/tIICQgICwkHBwM  
 CAgEBAwQFBgYIBQcGBgYHBgYKQCgGAWMBAP0n1QkJCQoJCAYDAGICAwQFBQcHBgYHBgYGBwYKQCg  
 GAWICAvGEAwYFAgEvAt0GBgUFBAMBafpe/NQKQKJCAGHBwYFBQQDAgICAgMEBQUGBwCICakJCQo  
 CUgEBBQcICGsGBwb6AQECawQFBQcGBwgICQkJAAAAAASAN4AAQAAAAAAAABAAAAAQAQAAAAAQA  
 NAAEAQAQAAAAAAgAHAA4AAQAAAAAAAwANABUAAQAAAAAABAANACIAAQAAAAAABQALAC8AAQAAAAA  
 ABgANADoAAQAAAAAACgAsAEcAAQAAAAAACwASAHMAAwABBAkAAAAACAIUAawABBAkAAQAaAICAAwA  
 BBakAAgAOAKEAAwABBAkAAwAaAK8AAwABBAkABAAaAAwABBAkABQAWAOMAawABBAkABgAaAPk  
 AAawABBAkACgBYARMAAwABBAkACwAkAWSwRmlsZVR5cGVfRm9udFJlZ3VsYXJGaWxlVHlwZV9Gb25  
 0RmlsZVR5cGVfRm9udFJlZ3VsYXJGaWxlVHlwZV9Gb250RmlsZVR5cGVfRm9udFJlZ3VsYXJGaWxlVHlwZV9Gb25  
 nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAARgBpAGwAZQBUAHk  
 AcABlAF8ARgBvAG4AdABSAGUAZwBlAGwAYQByAEYAAQBsAGUAVAB5AHAZQBfAEYABwBuAHQARgB  
 pAGwAZQBUAHkAcABlAF8ARgBvAG4AdABWAGUAcgBzAGkAbwBuACAAMQAuADAARgBpAGwAZQBUAHk  
 AcABlAF8ARgBvAG4AdABGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIAblAHMAaQBuaGCAIAAB  
 TAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHk  
 AbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
 AAAASAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgETAANCTVAEWEXtTWANBVkkDRE9

```
DA0VYRQNNUDQDWk1QA1RBUgRET0NYA0pQRwNQTKcDR01GA1BERgNUWFQES1BFRwNYTFMDTVazAAA
=) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'FileType_Font' !important;
speak: none;
font-size: 35px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
position: relative;
top: 8px;
}
/* csslint ignore:end */
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [Customize progress bar](#)
- [Customize button with HTML element](#)
- [Customize drop area](#)

## Localization in EJ2 JavaScript Uploader control

The Localization library allows you to localize static text content of the uploader. The static text contains default text content of action buttons, file status, clear icon title, tooltips, and text content of drag area. Define the [locale](#) object for a culture and assign it to L10n load method.

The following are the list of keys and its values used in the uploader component:

Key	Description
-----	-----
Browse	To customize the browse button text.
Clear	To customize the clear button text.
Upload	To customize the upload button text.
dropFilesHint	To customize the drop area text.
uploadFailedMessage	To customize the status text when the file is failed to upload.
uploadSuccessMessage	To customize the status text when the file is uploaded successfully.
removedSuccessMessage	To customize the status text when the file is removed the successfully from the serve.

- | removedFailedMessage | To customize the status text while the file is failed to remove. |
- | inProgress | To customize the status text while the upload is in progress. |
- | pauseUpload | To customize the status text while the uploading is paused. |
- | fileUploadCancel | To customize the status text when uploading is canceled. |
- | readyToUploadMessage | To customize the status text when the file is selected and ready to upload. |
- | invalidMaxFileSize | To customize the status text when the file size is greater than the maximum file size. |
- | invalidFileType | To customize the status text when the file type is invalid. |
- | invalidMinFileSize | To customize the status text when the file size is less than the minimum file size. |
- | remove | To customize tooltip text for remove icon. |
- | cancel | To customize tooltip text for cancel icon. |
- | delete | To customize tooltip text for delete icon. |
- | totalFiles | To customize tooltip text for total files. |
- | size | To customize tooltip text for size. |

## INDEX.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
import { Uploader } from '@syncfusion/ej2-inputs';
import { detach, L10n } from '@syncfusion/ej2-base';
L10n.load({
 "fr-CH": {
 "uploader": {
 "invalidMinFileSize" : "La taille du fichier est trop petite! S'il vous plaît télécharger des fichiers avec une taille minimale de 10 Ko",
 "invalidMaxFileSize" : "La taille du fichier dépasse 28 Mo",
 "invalidFileType" : "Le type de fichier n'est pas autorisé",
 "Browse" : "Feuilleter",
 "Clear" : "Clair",
 "Upload" : "Télécharger",
 "dropFilesHint" : "ou Déposer des fichiers ici",
 "uploadFailedMessage" : "Impossible d'importer le fichier",
 "uploadSuccessMessage" : "Fichier téléchargé avec succès",
 "removedSuccessMessage": "Fichier supprimé avec succès",
 "removedFailedMessage": "Le fichier n'a pas pu être supprimé",
 "inProgress": "Téléchargement",
 "readyToUploadMessage": "Prêt à télécharger",
 "remove": "Retirer",
 "cancel": "Annuler",
 "delete": "Supprimer le fichier",
 "totalFiles": "Total des fichiers",
 "size": "taille"
 }
 }
});
let dropElement: HTMLElement = document.getElementsByClassName('control-fluid')[0] as HTMLElement;
```

```
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 asyncSettings: {
 saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 dropArea: dropElement,
 locale: 'fr-CH'
});
uploadObj.appendTo('#fileupload');
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" class="fileupload">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## STYLES.CSS

```
#container {
 visibility: hidden;
```



```

margin: 0 auto;
width: 400px;
}
#loader {
color: #008cff;
font-family: 'Helvetica Neue', 'calibiri';
font-size: 14px;
height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Wai aria accessibility in EJ2 JavaScript Uploader control

The Uploader component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Uploader component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) |  |

| [Section 508 Support](#) |  |

| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The
component does not meet the requirement.</div>
```

### Keyboard interaction

The uploader component characterized with complete ARIA accessibility support that helps to be accessible by on-screen readers and other assistive technology devices.

The following are the standard keys that works on uploader component:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| **Tab** | Move focus to next element. |

| **Shift + Tab** | Move focus to previous element. |

| **Enter** | Triggers corresponding action to button element. |

| **Esc** | Close the file browser dialog alone and cancels the upload on drop the file. |

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize Uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
```

```
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container" class="fileupload">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### **STYLES.CSS**

```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Ensuring accessibility

The Uploader component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Uploader component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Uploader component with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

### Style appearance in EJ2 JavaScript Uploader control

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the appearance of File Upload wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
,
/ To specify height /
.e-upload.e-control-wrapper, .e-bigger.e-small .e-upload.e-control-wrapper {
height: 300px;
width: 300px;
}
,
```

#### Customizing the File Upload browse button

Use the following CSS to customize the File Upload browse button

```
,
/ To specify font size and color /
.e-upload .e-file-select-wrap .e-btn, .e-upload .e-upload-actions .e-btn, .e-bigger.e-small .e-upload .e-
file-select-wrap .e-btn, .e-bigger.e-small .e-upload .e-upload-actions .e-btn {
font-family: cursive;
height: 40px;
background-color: aquamarine;
color: coral;
}
,
```

#### Customizing the File Upload content

Use the following CSS to customize the File Upload content

```
,
```

*/ To specify font size and color /*

```
.e-upload .e-file-select-wrap .e-file-drop, .e-bigger.e-small .e-upload .e-file-select-wrap .e-file-drop {
font-size: 20px;
color: aqua;
}
,
```

### Customizing the uploaded file container in File Upload

Use the following CSS to customize the uploaded file container in File Upload

```
,
/ To specify background color /
.e-upload .e-upload-files .e-upload-file-list {
background-color: beige;
}
,
```

See Also

- [Customize the appearance of uploader using a template](#)

### How To

#### Hide default drop area in EJ2 JavaScript Uploader control

You can achieve this behavior by overriding the corresponding uploader styles. Override the following styles to hide the default drop area behavior.

- .e-upload.e-control
- .e-upload .e-file-select
- .e-upload .e-file-drop

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 Uploader</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Uploader Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### **STYLES.CSS**

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.e-control {
 border : 0;
}

```

```
.e-control ul {
 border: 1px solid #ddd;
}
.e-control .e-file-select,
.e-control .e-file-drop {
 display: none;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Preview images before uploading in EJ2 JavaScript Uploader control

The uploader component allows to create preview images before uploading. The preview images can be created by reading the file using selected event. Also, the user can create preview images after uploading to server using success event. Refer to the following link to learn about how to create image preview.

### [Image Preview](#)

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Add html attributes in EJ2 JavaScript Uploader control

You can add the additional HTML attributes such as disabled, value, name, and more to the element using the [htmlAttributes](#) property. If you configured both the property and equivalent HTML attribute, then the component considers the property value.

The following example demonstrates how to set attributes in htmlAttributes property in the Uploader.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
//Initialize the control by preload files
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 htmlAttributes: { "disabled": "true" }
});
uploadObj.appendTo('#fileupload');
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}

#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

.e-control {
 border : 0;
}

.e-control ul {
 border: 1px solid #ddd;
}

```



```
.e-control .e-file-select,
.e-control .e-file-drop {
 display: none;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Achieve invisible upload in EJ2 JavaScript Uploader control

You can achieve the invisible upload feature by using the selected event in uploader component. Refer to the following example.

#### INDEX.TS

```
import { Uploader, SelectedEventArgs } from '@syncfusion/ej2-inputs';
import { createElement } from '@syncfusion/ej2-base';
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 selected : onupload,
 locale: 'en-US',
 allowedExtensions: '.png, .jpg, .jpeg'
});
uploadObj.appendTo('#fileupload')
function onupload(args: SelectedEventArgs){
 for(let i = 0; i< args.filesData.length ; i++){
 let liparentDiv = createElement('div', { className: 'image-list'});
 let liImage = createElement('img', { className: 'image'});
 liparentDiv.appendChild(liImage);
 readURL(liImage, args.filesData[i]);
 document.getElementById('preview').appendChild(liparentDiv);
 }
 args.cancel=true;
}
function readURL(liImage: HTMLElement, file: any) {
 let imgPreview: HTMLImageElement = liImage as HTMLImageElement;
 let imageFile: File = file.rawFile;
 let reader: FileReader = new FileReader();
 reader.addEventListener('load', () => {
 imgPreview.src = reader.result;
 }, false);
 if (imageFile) {
 reader.readAsDataURL(imageFile);
 }
}
};
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

<title>Essential JS 2 Uploader</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Uploader Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="preview"></div>
 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### **STYLES.CSS**

```

#container {
 visibility: hidden;
 margin: 10px;
}
#loader {
 color: #008c00;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
#preview {
 border: 2px dashed #ddd;
 padding: 15px;
}

```

```
.image-list {
 width: 134px;
 height: 117px;
 margin-right: 4px;
 border: 1px solid lightgrey;
 display: inline-block;
}
.image {
 width: 134px;
 height: 117px;
 margin-right: 4px;
 display: inline-block;
}
.e-control {
 border : 0;
}
.e-control ul {
 border: 1px solid #ddd;
}
.e-control .e-file-select,
.e-control .e-file-drop {
 display: none;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Customize progressbar in EJ2 JavaScript Uploader control

You can customize the progress bar's size, color, and background by overriding the styles in uploader component. Refer to the following example.

#### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
// initialize uploader component
let uploadObject: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 }
});
// render initialized Uploader
uploadObject.appendTo('#fileupload');
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 300px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
#container .e-control .e-upload-files .e-file-container .e-progress-inner-
wrap .e-upload-progress {
 background: yellow;
 height: 4px;
}
#container .e-control .e-upload-files .e-file-container .e-progress-inner-
wrap {
 height: 4px;
 background-color: lightblue;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Sort the selected files in EJ2 JavaScript Uploader control

You can sort the selected files in uploader component by using the [selected](#) event. Refer to the following example.

#### INDEX.TS

```
import { Uploader, FileInfo, SelectedEventArgs } from '@syncfusion/ej2-inputs';
import { Event } from '@syncfusion/ej2-base';
let initial: boolean = true;
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 selected: onSelect
});
uploadObj.appendTo('#fileupload')
function onSelect(args: SelectedEventArgs): void {
 if (initial) { initial = false; return; }
 args.isModified = true;
 let oldFiles: FileInfo[] = uploadObj.GetFilesData();
 let filesData: FileInfo[] = args.filesData.concat(oldFiles);
 let modifiedData: FileInfo[] = sortFileList(filesData);
 args.modifiedFilesData = modifiedData;
}
function sortFileList(filesData: FileInfo[]): FileInfo[] {
 let files: FileInfo[] = filesData;
 let fileNames: string[] = [];
 for (let i: number = 0; i < files.length; i++) {
 fileNames.push(files[i].name);
 }
 let sortedFileNames: string[] = fileNames.sort();
 let sortedFilesData: FileInfo[] = [];
 let index: number = 0;
 for (let name of sortedFileNames) {
 for (let i: number = 0; i < files.length; i++) {
 if (name === files[i].name) {
 sortedFilesData.push(files[i]);
 }
 }
 }
 return sortedFilesData;
}
```

#### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Essential JS 2 Uploader Component">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 300px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

Get the total size of selected files in EJ2 JavaScript Uploader control

You can get the total size of selected files before uploading it to the designated server. This can be achieved by using the selected event. Refer to the following example to calculate the total file size.

### INDEX.TS

```
import { Uploader, FileInfo, SelectedEventArgs } from '@syncfusion/ej2-inputs';
import { Event } from '@syncfusion/ej2-base';
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 selected: onSelect
});
uploadObj.appendTo('#fileupload')
function onSelect(args: SelectedEventArgs): void {
 let totalSize: number = 0;
 for (let file of args.filesData) {
 totalSize = totalSize + file.size;
 }
 let size: string = uploadObj.bytesToSize(totalSize);
 alert("Total select file's size is " + size);
}
```

### INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
 type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
 ="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>
```

```
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

### STYLES.CSS

```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 300px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Customize button with html element in EJ2 JavaScript Uploader control

The uploader component allows you to customize the action buttons by using [buttons](#) property. Refer to the following example.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
import { createElement } from '@syncfusion/ej2-base';
let uploadEle = createElement('span', { className: 'upload e-icons' });
uploadEle.innerHTML = 'Upload All';
let clearEle = createElement('span', { className: 'remove e-icons' });
clearEle.innerHTML = 'Clear All';
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 autoUpload: false,
 buttons: {
 browse: 'Choose file',
 clear: clearEle,
 upload: uploadEle
 }
});
```



```
});
uploadObj.appendTo('#fileupload')
```

## INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>
```

## STYLES.CSS

```
#container {
 visibility: hidden;
 margin: 0 auto;
 width: 450px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
```

```

width: 30%;
}
.e-upload .e-upload-actions .e-file-clear-btn,
.e-upload .e-upload-actions .e-file-upload-btn {
 -webkit-tap-highlight-color: transparent;
 background-color: #ddd;
 border-color: #c3c3c3;
 color: #000000;
 box-shadow: none;
 width: 44%;
 margin: 0;
 text-transform: none;
 padding: 8px 0;
}
.e-upload .e-upload-actions {
 width: 100%;
}
.upload::before {
 content: '\e725';
 position: relative;
 font-size: 10px;
 right: 10px;
}
.remove::before {
 content: '\e932';
 position: relative;
 font-size: 10px;
 right: 10px;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

Add confirm dialog to remove the files in EJ2 JavaScript Uploader control

You can customize the uploader component using confirm dialog before removing the files. Here, ej2 dialog is used as confirm dialog. Refer to the following example.

#### INDEX.TS

```

import { Uploader, FileInfo } from '@syncfusion/ej2-inputs';
import { createElement } from '@syncfusion/ej2-base';
import { Dialog } from '@syncfusion/ej2-popups';
let removeFile: FileInfo[];
let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 removing: onremove
});
uploadObj.appendTo('#fileupload');
// Initialize Dialog component
let dialog: Dialog = new Dialog({

```

```

 content: 'Are you sure want to remove the file?',
 buttons: [{ 'click': () => { onClick() }, buttonModel: { content: 'OK',
cssClass: 'e-flat', isPrimary: true }},
 { 'click': () => { dialog.hide(); }, buttonModel: { content: 'Cancel',
cssClass: 'e-flat' } }],
 width: '250px',
 visible: false,
 target: '#container'
 });
 dialog.appendTo('#dialog');
 function onClick() {
 dialog.hide();
 uploadObj.remove(removeFile[0], false, true);
 }
 function onremove(args: any) {
 removeFile=[];
 args.cancel = true;
 removeFile.push(args.filesData);
 dialog.show();
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 <div id="dialog"></div>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}

```

```

 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

html,
body,
#container {
 height: 100%;
 overflow: hidden;
 margin: auto 30px;
 padding: 20px 0;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Add additional data on upload in EJ2 JavaScript Uploader control

The uploader component allows you to add additional data on file upload, which is used to get in the server-side. By using [uploading](#) event and its customFormData argument, you can achieve this behavior. Refer to the following example.

In the following code snippet, explains about how to add additional data on file upload.

```

`ts
import { Uploader } from '@syncfusion/ej2-inputs';

let uploadObj: Uploader = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 uploading: onFileUpload
});

uploadObj.appendTo('#fileupload');

function onFileUpload(args: any) {

```

```
// add additional data as key-value pair.
```

```
args.customFormData = [{ 'name': 'Syncfusion INC' }];
}
`
```

*Server side for adding additional data*

```
`c#
// Get the additional data in server end by corresponding key.
var data = HttpContext.Current.Request.Form["name"];
`
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Validate image on drop in EJ2 JavaScript Uploader control

The uploader component allows you to upload all type of images by setting **image/\*** to [allowedExtensions](#) property.

By default, the behavior is working with select a file using browse button. But, this behavior doesn't support on drag and drop the files. You can handle this behavior manually using **selected** event by filtering the file types from application.

In the following example, validated image files using images/\*. You are able to drag and drop the image files with extension of PNG, JPG, BPG, GIF and TIFF to upload it.

### INDEX.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
let uploadObj: any = new Uploader({
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 autoUpload: false,
 allowedExtensions: 'image/*',
 selected: onSelect
});
uploadObj.appendTo('#fileupload');
function onSelect(args: any): void {
 if (args.event.type === 'drop') {
 let allImages: Array<string> = ['png', 'jpg', 'jpeg', 'gif', 'tiff',
'bpg'];
 let files = args.filesData;
 let modifiedFiles = [];
 for (let file of files) {
 if (allImages.indexOf(file.type) === -1) {
 file.status = 'File type is not allowed';
 file.statusCode = '0';
 }
 modifiedFiles.push(file);
 }
 }
}
```

```

 args.isModified = true;
 args.modifiedFilesData = modifiedFiles.concat(uploadObj.filesData);
 }
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## STYLES.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;

```

```

top: 45%;
width: 30%;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Determine whether the uploader has input file in EJ2 JavaScript Uploader control

By setting the **required** attribute to uploader input element, you can validate the input file that has any value in it. In the following sample, set required attribute to the uploader input element and showcase the validation failure message using the **data-required-message** attribute.

### INDEX.TS

```

import { detach, createElement } from '@syncfusion/ej2-base';
import { FormValidator, Uploader } from '@syncfusion/ej2-inputs';
import { Dialog } from '@syncfusion/ej2-popups';
let dropElement = document.getElementsByClassName('dropUpload')[0];
// Initialize the uploader component
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 selected: onFileSelect,
 allowedExtensions: 'image/*',
 multiple: true,
 dropArea: dropElement,
});
uploadObj.appendTo('#fileupload');
document.getElementById('customBrowse').onclick = () => {
 document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button').click();
};
function onFileSelect(args: any): void {
 if (args.filesData.length > 0) {
 if (document.getElementsByClassName('upload-image').length > 0) {
 detach(document.getElementsByClassName('imgWrapper')[0]);
 }
 let imageTag = createElement('IMG', { className: 'upload-image',
attrs: { 'alt': 'Image' } });
 let wrapper: HTMLElement = createElement('span', { className:
'imgWrapper' }) as HTMLElement;
 wrapper.appendChild(imageTag);
 let rootFile = document.getElementsByClassName('dropUpload')[0];
 rootFile.insertBefore(wrapper, rootFile.firstChild);
 readURL(wrapper, args.filesData[0]);
 }
 args.cancel = true;
}
function readURL(li: HTMLElement, args: any): void {
 let preview: HTMLImageElement = li.querySelector('.upload-image');
 let file: File = args.rawFile; let reader: FileReader = new
FileReader();
 reader.addEventListener('load', () => { preview.src = reader.result; },
false);
 if (file) { reader.readAsDataURL(file); }
}

```

```

}
let options = {};
let formObj: FormValidator = new FormValidator('#form1', options);
function onFormSubmit(): void {
 let formStatus: Boolean = formObj.validate();
 if (formStatus) {
 formObj.element.reset();
 detach(document.getElementsByClassName('imgWrapper')[0]);
 confirm.show();
 }
}
let confirm: Dialog = new Dialog({
 width: '335px',
 visible: false,
 header: 'Success',
 content: 'Your details have been updated successfully, Thank you.',
 target: document.getElementById('control_wrapper'),
 showCloseIcon: true,
 isModal: true,
 animationSettings: {
 effect: 'Zoom'
 }
});
confirm.appendTo('#confirmationDialog');
document.getElementById('submit-btn').onclick = () => {
 onFormSubmit();
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="col-lg-12 control-section">

```



```

 <h4 class="form-title">Photo Contest</h4>
 <div class="control_wrapper" id="control_wrapper">
 <form id="form1" method="post">
 <div class="form-group" style="padding-top: 40px; float:
left">
 <div class="e-float-input">
 <input type="text" id="name" name="name" data-
required-message="* Enter your name" required="" data-msg-
containerid="nameError">

 <label class="e-float-text e-label-top"
for="name">Name</label>
 </div>
 <div id="nameError"></div>
 </div>
 <div id="dropArea">
 <div id="uploadError" style="float: right;"></div>
 <div id="customBrowse" class="form-group dropUpload">
Drop image here...
 <input type="file" name="UploadFiles"
id="fileupload" data-required-message="* Choose your image to upload"
required="" data-msg-containerid="uploadError">
 </div>
 </div>
 <div class="submitBtn">
 <button type="button" class="submit-btn e-btn"
id="submit-btn">Submit</button>
 <div class="desc">*This button is not a submit
type and the form submit handled from externally.</div>
 </div>
 </form>
 <div id="confirmationDialog"></div>
 </div>
 </div>
 </div>
 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
 </body></html>

```

## STYLES.CSS

```

#container {
 visibility: hidden;
 margin: 0 auto;
 width: 400px;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
}

```

```

height: 40px;
left: 45%;
position: absolute;
top: 45%;
width: 30%;
}
.control_wrapper {
max-width: 400px;
margin: auto;
}
.control_wrapper .e-upload .e-upload-drag-hover {
margin: 0;
}
.address-field {
resize: none;
}
#dropArea .dropUpload .upload-image {
height: 140px;
width: 140px;
}
#dropArea .dropUpload {
float: right;
text-align: center;
vertical-align: middle;
line-height: 12;
overflow: hidden;
border: 1px dashed;
width: 150px;
height: 150px;
}
.e-upload {
visibility: hidden;
}
#control_wrapper {
max-width: 500px;
margin: auto;
border: 0.5px solid #BEBEBE;
box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
padding: 1% 4% 7%;
background: #f9f9f9;
}
.e-error {
padding-top: 3px;
}
.control_wrapper .e-upload .e-upload-drag-hover {
margin: 0;
}
.submit-btn {
margin-top: 15px;
margin: auto 100px;
}
.submitBtn .desc {
margin: 2% 23% 0 18%;
}
.submitBtn {
text-align: center;
}

```

```

.form-support {
 width: 100%;
}
.success .form-support {
 display: none;
}
.success .successmsg {
 border: 0.5px solid green;
 padding: 10%;
 color: green;
}
#form1 {
 position: relative;
 top: 14%;
}
.form-support td {
 width: 100%;
 padding-top: 4%;
}
.e-upload {
 float: none;
}
.choose-file {
 width: 60%;
}
#browse {
 float: right;
 margin-right: -113px;
 margin-top: -27px;
}
.form-title {
 text-align: center;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

Achieve file upload programmatically in EJ2 JavaScript Uploader control

You can upload a file programmatically using the [upload](#) method. Get the selected files data from the [getFilesData](#) public method in uploader.

The upload method behaves differently based on its arguments.

- If this method receives any files as arguments, those files only start to upload.
- If it has no argument, all the selected files start to upload.

## INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
//Initialize the control by preload files
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',

```

```

 removeUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 }
});
uploadObj.appendTo('#fileupload');
document.getElementById('first').onclick = (args) => {
 uploadObj.upload(uploadObj.getFilesData()[0]);
};
document.getElementById('full').onclick = (args) => {
 uploadObj.upload(uploadObj.getFilesData());
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="control_wrapper">
 <!-- Initialize Uploader -->
 <input type="file" name="UploadFiles" id="fileupload">
 </div>

 <button id="first" class="e-btn e-control">Upload first
file</button>

 <button id="full" class="e-btn e-control">Upload all
files</button>

 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {

```

```

 ele.style.visibility = "visible";
 }
 </script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.control_wrapper {
 max-width: 500px;
 margin: auto;
}
.e-upload {
 width: 100%;
 position: relative;
 margin-top: 15px;
 margin-bottom: 15px;
}
.e-upload-actions {
 display: none;
}
.e-btn {
 text-transform: none;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Check file size before uploading in EJ2 JavaScript Uploader control

By using the [uploading](#) event, you can get the file size before uploading it to the server.

File object contains the file size in bytes only. You can convert the size to standard formats (KB or MB) using [bytesToSize](#) method.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
//Initialize the control by preload files
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 asyncSettings: {

```

```

 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 uploading: onBeforeUpload
});
uploadObj.appendTo('#fileupload');
function onBeforeUpload(args): void {
 // get the file size in bytes
 let sizeInBytes: number = args.fileData.size;
 // get the file size in standard format
 alert("File size is: " + uploadObj.bytesToSize(sizeInBytes));
}

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

## STYLES.CSS

```
#container {
```

```

 visibility: hidden;
 }
 #loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

Check the mime type of file before upload in EJ2 JavaScript Uploader control

By using the [uploading](#) event, you can get the file MIME type before uploading it to the server. In the following sample, file MIME type is shown in the alert box before the file starts to upload.

#### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
//Initialize the control by preload files
let uploadObj: Uploader = new Uploader({
 autoUpload: false,
 asyncSettings: {
 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 uploading: onBeforeUpload
});
uploadObj.appendTo('#fileupload');
function onBeforeUpload(args): void {
 // get the file MIME type
 alert("File MIME type is: " + args.fileData.rawFile.type)
}

```

#### INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">

```

```

<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <input type="file" id="fileupload" name="UploadFiles">
 </div>

<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

### STYLES.CSS

```

#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

Trigger click event of input file from external button in EJ2 JavaScript Uploader control

Click event of input file from the external button can be triggered using the `click` event of button. In the following sample, you can find the triggered click event of input file from `Essential JavaScript 2 Button`.

### INDEX.TS

```

import { Uploader } from '@syncfusion/ej2-inputs';
//Initialize the control by preload files
let uploadObj: Uploader = new Uploader({
 asyncSettings: {

```



```

 saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
 removeUrl:
 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
 },
 });
uploadObj.appendTo('#fileupload');
document.getElementById('browse').onclick = () => {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
};

```

## INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
 <title>Essential JS 2 Uploader</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta name="description" content="Essential JS 2 Uploader Component">
 <meta name="author" content="Syncfusion">
 <link href="index.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
inputs/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
 <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">

 <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
 <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

 <div id="container">
 <div class="control_wrapper">
 <!-- Initialize Uploader -->
 <div id="dropArea">
 Drop image (JPG, PNG) files here or
<button class="e-btn e-control" id="browse">Browse</button>
 </div>
 <input type="file" name="UploadFiles" id="fileupload">
 </div>
 </div>

 <script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
 </script>
 <script src="index.js" type="text/javascript"></script>
</body></html>

```

**STYLES.CSS**

```
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 font-family: 'Helvetica Neue', 'calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.control_wrapper {
 max-width: 500px;
 margin: auto;
}
#dropArea {
 border: 1px dashed #c3c3cc;
 text-align: center;
 padding: 20px 0 10px;
}
.e-file-select-wrap {
 display: none;
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Open and edit the uploaded files in EJ2 JavaScript Uploader control

The uploader component allows you to modify the file after uploading to the server, which can be achieved using the `success` event of the uploader.

You can retrieve the saved file path in the uploader success event and assign it to custom attribute (data-file-name) value of the respective file list element to open the uploaded file. Click the respective file element to create a new request along with saved file path using the HTTP header. In the server-side, get the file path from the header and open the file using `process.start` method.

```
`ts
```

```
import { Uploader } from '@syncfusion/ej2-inputs';

let uploadObj: any = new Uploader({
 asyncSettings: {
 saveUrl: 'SaveUrl',
 removeUrl: 'RemoveUrl'
 },

```

```

success: onUploadSuccess
});
uploadObj.appendTo('#fileupload');
function onUploadSuccess(args) {
// fetching the generated li elements
var liElements = this.uploadWrapper.querySelectorAll('.e-upload-file-list');
for (var i = 0; i < liElements.length; i++) {
if (liElements[i].getAttribute('data-file-name') == args.file.name) {
liElements[i].addEventListener('click', () => { openFile(args, event) })
// File path have to update from server end in response status description.
liElements[i].setAttribute('file-path', args.e.target.statusText);
}
}
}
function openFile(args, e) {
if (!e.target.classList.contains('e-file-delete-btn') && !e.target.classList.contains('e-file-remove-btn'))
{
let ajax = new XMLHttpRequest();
// create new request for open the selected file
ajax.open("POST", '/Home/openFile');
var liElements = document.getElementsByClassName('e-upload')[0].querySelectorAll('.e-upload-file-list');
for (var i = 0; i < liElements.length; i++) {
if (liElements[i].getAttribute('data-file-name') == args.file.name) {
// Added the file path in header to get it in server side.
ajax.setRequestHeader('filePath', liElements[i].getAttribute('file-path').toString());
}
}
ajax.send();
}
}
,

```

*Server side for open and edit the uploaded files*

`c#

```

public void Save() {
 if (!System.IO.File.Exists(fileSavePath))
 {
 httpPostedFile.SaveAs(fileSavePath);
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 // Sending the file path to client side
 Response.StatusDescription = fileSavePath;
 Response.End();
 }
}

[AcceptVerbs("Post")]
public void openFile()
{
 // Check whether the file is available in the corresponding location
 if (System.IO.File.Exists(Request.Headers.GetValues("filePath").First()))
 {
 // This will open the selected file from server location in desktop
 Process.Start(Request.Headers.GetValues("filePath").First());
 }
}

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Resize images before uploading it to the server in EJ2 JavaScript Uploader control

You can customize the dimension of the images before uploading it to the server. By using selected event, you can get the selected file information as type of an object. From the obtained image file information, create a new canvas and render an image with the custom dimensions. Refer the corresponding code snippet as follows.

```

<div id='upload'>
<!-- Initialize Uploader -->
<input type='file' id='customUI' name='UploadFiles' />

```

```
</div>
<style>
.upload-success {color: #2bc700;}
.upload {
width: 450px;
margin: auto 400px;
}
.e-upload {
width: 100%;
margin-top: 50px;
}
.upload-list-root {
display: inline-block;
width: 450px;
}
.ul-element {
list-style: none;
padding-left: 0px;
width: 100%;
}
.file-name-container {
width: 100%;
}
.file-name {
padding: 8px 6px 8px 4px;
font-size:13px;
width:100px;
display: inline-block;
text-overflow: ellipsis;
overflow: hidden;
white-space: nowrap;
position: relative;
top: 2px
```

```
}
.file-size {
padding: 4px 0px;
font-size: 13px;
display: initial;
position: relative;
top: -8px;
}
.file-lists {
border: 1px solid lightgray;
padding: 0px 6px 0px 14px;
margin-top: 15px;
position: relative;
background: rgba(0, 0, 0, 0.04);
}
span.file-status-container {
padding: 10px 26px;
font-size: 13px;
position: relative;
top: 0px;
right: 60px;
float: right;
}
.file-status-container, .file-size, .file-name {
font-family: "Helvetica Neue", "Helvetica", "Arial", "sans-serif";
}
span.progress-bar-container {
display: block;
float: right;
height: 20px;
right: 50px;
top: 11px;
position: relative;
```

```
right: 107px;
width: 139px
}
.progress{
width: 100%;
height: 15px;
-webkit-appearance: none;
}
.file-status-container {
display: none;
}
.close-icon-container
{
cursor: pointer;
font-size: 11px;
height: 24px;
margin: 0 12px;
padding: 0;
position: absolute;
right: 0;
width: 24px;
top: 8px;
}
.close-icon-container.e-icons::before {
left: 6px;
position: inherit;
top: 6px;
content: '\e932';
}
progress::-webkit-progress-bar {
border: 1px solid lightgrey;
background-color: #ffffff;
border-radius: 2px;
```

```

}
progress::-webkit-progress-value {
border-radius: 2px;
background-color: skyblue;
}
</style>
`ts
/

```

- Pre-resized image before upload

```

*/
import { Uploader, FileInfo } from '@syncfusion/ej2-inputs';
import { detach, isNullOrUndefined, createElement, EventHandler } from '@syncfusion/ej2-base';
let uploadObj: Uploader = new Uploader({
autoUpload: false,
asyncSettings: {
saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
},
allowedExtensions: 'image/*',
selected: onFileSelect,
progress : onFileUpload,
success : onuploadSuccess,
failure : onuploadFailed
});
uploadObj.appendTo('#customUI');
let newImage;
let parentElement : HTMLElement;
let ul: HTMLElement;
let proxy : any;
function onFileSelect(args : any) : void {
args.cancel = true;

```



```
proxy = this;
if (isNullOrUndefined(document.getElementById('upload').querySelector('.upload-list-root'))) {
 parentElement = createElement('div', { className: 'upload-list-root' });
 ul = createElement('ul', {className: 'ul-element' });
 parentElement.appendChild(ul);
 document.getElementById('upload').appendChild(parentElement);
}
for (let i : number = 0; i < args.filesData.length; i++) {
 formSelectedData(args.filesData[i]);
}
this.filesData = this.filesData.concat(args.filesData);
let file: FileInfo = args.filesData[0].rawFile;
let width: number;
let height: number;
let img: any = document.createElement("img");
let reader: any = new FileReader();
reader.onload = function(e: any) { img.src = e.target.result; };
reader.readAsDataURL(file);
let imgs = new Image();
img.onload = function() {
 width = this.width;
 height = this.height;
 onNewImg(height, width, img, args.filesData[0])
}
imgs.src = img.src;
}
// to create canvas and update our custom dimensions
function onNewImg(height: any, width: any, img: any, file: any) {
 let canvas: HTMLCanvasElement = document.createElement("canvas");
 let ctx: any = canvas.getContext("2d");
 ctx.drawImage(img, 0, 0);
 let MAX_WIDTH: any = 1000;
 let MAX_HEIGHT: any = 600;
```

```
if (width > height) {
 if (width > MAX_WIDTH) {
 height *= MAX_WIDTH / width;
 width = MAX_WIDTH;
 }
} else {
 if (height > MAX_HEIGHT) {
 width *= MAX_HEIGHT / height;
 height = MAX_HEIGHT;
 }
}

canvas.width = width;
canvas.height = height;
let ctx1 = canvas.getContext("2d");
ctx1.drawImage(img, 0, 0, width, height);
newImage = canvas.toDataURL("image/png");
let blobBin = atob(newImage.split(',')[1]);
let array = [];
for(var i = 0; i < blobBin.length; i++) {
 array.push(blobBin.charCodeAt(i));
}

let newBlob = new Blob([new Uint8Array(array)], {type: 'image/png'});
let newFile: any = createFile(newBlob, file);
uploadObj.upload(newFile, true);
}

// To create File object to upload
function createFile(image: any , file: any) {
 let newFile = {
 name: file.name,
 rawFile: image,
 size: image.size,
 type: file.type,
 validationMessage: "",
```

```
statusCode: '1',
status: 'Ready to Upload'
}
return newFile;
}

function formSelectedData (selectedFiles : FileInfo) : void {
let liEle : HTMLElement = createElement('li', { className: 'file-lists', attrs: {'data-file-name' :
selectedFiles.name} });
let fileNameContainer : HTMLElement = createElement('span', {className: 'file-name-container' });
let fileName : HTMLElement = createElement('span', {className: 'file-name ' });
fileName.innerHTML = selectedFiles.name;
let fileSize : HTMLElement = createElement('span', {className: 'file-size ' });
fileSize.innerHTML = proxy.bytesToSize(selectedFiles.size);
liEle.appendChild(fileName);
liEle.appendChild(fileSize);
let fileStatusContainer : HTMLElement = createElement('span', {className: 'file-status-container' });
let fileStatus : HTMLElement = createElement('span', {className: 'file-status'});
fileStatusContainer.appendChild(fileStatus);
let progressbarContainer : HTMLElement = createElement('span', {className: 'progress-bar-container'});
let progressBar : HTMLElement = createElement('progress', {className: 'progress', attrs: {value : '0',
max : '100'}});
progressbarContainer.appendChild(progressBar);
let closelconContainer : HTMLElement = createElement('span', {className: 'e-icons close-icon-
container'});
EventHandler.add(closelconContainer, 'click', removeFiles, proxy);
liEle.appendChild(fileStatusContainer);
liEle.appendChild(progressbarContainer);
liEle.appendChild(closelconContainer);
ul.appendChild(liEle);
proxy.fileList.push(liEle);
}

function onFileUpload(args : any) : void {
let li : Element = document.getElementById('upload').querySelector('[data-file-name="" + args.file.name
+ "']');
```

```
let progressValue : number = Math.round((args.e.loaded / args.e.total) * 100);
li.getElementsByTagName('progress')[0].value = progressValue;
}

function onuploadSuccess(args : any): void {
let li : Element = document.getElementById('upload').querySelector('[data-file-name="" + args.file.name
+ "']');
if (!NullOrUndefined(li.querySelector('.progress-bar-container')) {
detach(li.querySelector('.progress-bar-container'));
}

if (args.operation === 'upload') {
li.querySelector('.file-status').classList.add('upload-success');
li.querySelector('.file-status').innerHTML = args.file.status;
li.querySelector('.file-status-container').setAttribute('style', 'display: inline-block');
}

if (args.operation === 'remove') {
let index: number = this.fileList.indexOf(li);
this.fileList.splice(index, 1);
this.filesData.splice(index, 1);
detach(li);
}
}

function onuploadFailed(args : any): void {
let li : Element = document.getElementById('upload').querySelector('[data-file-name="" + args.file.name
+ "']');
li.querySelector('.file-status').innerHTML = args.file.status;
if (args.operation === 'upload') {
detach(li.querySelector('.progress-bar-container'));
li.querySelector('.file-status-container').setAttribute('style', 'display: initial');
}
}

function removeFiles(args : any): void {
let index: any = proxy.fileList.indexOf(args.currentTarget.parentElement);
return proxy.remove(proxy.filesData[index]);
}
```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Convert image into binary format after uploading in EJ2 JavaScript Uploader control

By default, the file upload component saves the uploaded image files in physical directories. Also, you can convert the images into binary format at server-side before saving the uploaded images. To retrieve binary format of image files, convert the posted file's input stream into binary reader and read as bytes using ReadBytes method.

Refer to the below server-side code snippet

```
`c#
[AcceptVerbs("Post")]
public void Save()
{
 try
 {
 if (System.Web.HttpContext.Current.Request.Files.AllKeys.Length > 0)
 {
 var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
 if (httpPostedFile != null)
 {
 byte[] fileBytes;
 using (BinaryReader br = new BinaryReader(httpPostedFile.InputStream))
 {
 fileBytes = br.ReadBytes((int)httpPostedFile.InputStream.Length);
 // bytes will be stored in variable fileBytes
 }
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 200;
 Response.Status = "200 Success";
 Response.End();
 }
 }
 }
}
```

```

}
catch (Exception e)
{
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 204;
 Response.Status = "204 No Content";
 Response.StatusDescription = e.Message;
 Response.End();
}
}
,

```

You can also explore [JavaScript File Upload](#) feature tour page for its groundbreaking features. You can also explore our [JavaScript File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Ej1 api migration in EJ2 JavaScript Uploader control

This article describes the API migration process of File Upload component from Essential JS 1 to Essential JS 2.

#### Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Localization	Property: locale  \$('#uploadBox').ejUploadbox({ locale: 'es-ES' })	Property: locale  var uploader =new ej.inputs.Uploader({ locale: 'es-ES' }) uploader.appendTo('#ej2_Uploader')
Right to left	Property: enableRTL  \$('#uploadBox'). ejUploadbox({ enableRTL: true }) uploader.appendTo('#ej2_Uploader')	Property: enableRtl  var uploader =new ej.inputs.Uploader({ enableRtl: true }) uploader.appendTo('#ej2_Uploader')

### File list

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Show/Hide the selected files	Property: showFileDetails  \$('#uploadBox').ejUploadbox({ showFileDetails: false })	Property: showFileList  var uploader =new ej.inputs.Uploader({ showFileList: false }) uploader.appendTo('#ej2_Uploader')
Customizing the file list	Not Applicable	Property: template  var uploader =new ej.inputs.Uploader({ template: '#templateId' }) uploader.appendTo('#ej2_Uploader')
Get the files in sorted form	Not Applicable	Method: sortFileList  var uploader =new ej.inputs.Uploader() uploader.appendTo('#ej2_Uploader') uploader.sortFileList(files)
Clearing File List	Not Applicable	Method: clearAll  var uploader =new ej.inputs.Uploader() uploader.appendTo('#ej2_Uploader') uploader.clearAll()
Event triggers when clearing Files	Not Applicable	Event : clearing  var uploader =new ej.inputs.Uploader({ clearing: function() {} }) uploader.appendTo('#ej2_Uploader')

### File selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Select multiple files to upload	Property: multipleFilesSelection  \$('#uploadBox').ejUploadbox({multipleFilesSelection: true})	Property: multiple  var uploader =new ej.inputs.Uploader({multiple: true}) uploader.appendTo('ej2_uploadbox')
Set minimum file size to upload	Not Applicable	Property: minFileSize  var uploader =new ej.inputs.Uploader({minFileSize: 1024}) uploader.appendTo('ej2_uploadbox')
Set maximum file size to upload	Property: fileSize  \$('#uploadBox').ejUploadbox({fileSize: 5000})	Property: maxFileSize  var uploader =new ej.inputs.Uploader({maxFileSize: 5000}) uploader.appendTo('ej2_uploadbox')
Allowed file types to select	Property: extensionsAllow  \$('#uploadBox').ejUploadbox({extensionsAllow: '.zip'})	Property: allowedExtensions  var uploader =new ej.inputs.Uploader({allowedExtensions: '.pdf'}) uploader.appendTo('ej2_uploadbox')
Restricted files types to select	Property:extensionsDeny  \$('#uploadBox').ejUploadbox({extensionsDeny: '.docx'})	Not Applicable
Display only selected details in File list	Property: customFileDetails  \$('#uploadBox').ejUploadbox({customFileDetails: { title: false, name: true, size: true, status: true, action: false } })	Not Applicable
Options to customize File list dialog	Property:dialogAction  \$('#uploadBox').ejUploadbox({dialogAction: { modal: false, closeOnComplete: true, resize: true, drag: false,	Not Applicable



	content: '#dialogTarget' } })	
Customize dialog position	Property:dialogPosition  \$('#uploadBox').ejUploadbox({ dialogPosition: {X: 300, Y 100} })	Not Applicable
Change file list key values	Property:dialogText  \$('#uploadBox').ejUploadbox({ dialogText: { title: 'Upload File List', name: 'File Name', size: 'File Size' } })	Not Applicable
Change drop area text	Property:dropAreaText  \$('#uploadBox').ejUploadbox({ dropAreaText: 'Drop files here' })	No separate property to change dropAreaText.It can be customize using locale Texts
Change drop area height	Property:dropAreaHeight  \$('#uploadBox').ejUploadbox({ dropAreaHeight: '100%' })	Not Applicable
Change drop area width	Property:dropAreaWidth  \$('#uploadBox').ejUploadbox({ dropAreaWidth: '100%' })	Not Applicable
Dynamically push the file	Property:pushFile  \$('#uploadBox').ejUploadbox({ pushFile: files })	Not Applicable
Show the files uploader in server already.	Not Applicable	Property: files  var preloadFiles = [{ name: 'nature', size: 5000, type: '.png' }] var uploader =new ej.inputs.Uploader({ files: preloadFiles }) upload.appendTo('#ej2_uploadBox')

Event triggers when select the file successfully	Event: fileSelect <pre>\$('#uploadBox').ejUploadbox({   fileSelect: function() {} })</pre>	Event: selected <pre>var uploader =new ej.inputs.Uploader({   selected: onFileSelect }) upload.appendTo('#ej2_uploadBox')function onFileSelect() {}</pre>
--------------------------------------------------	-----------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

### Upload action

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Save URL	Property: saveUrl <pre>\$('#uploadBox').ejUploadbox(   ({     saveUrl: 'saveUrl'   }) )</pre>	Property: asyncSettings.saveUrl <pre>var uploader =new ej.inputs.Uploader({   asyncSettings: { saveUrl: 'saveUrl' } }) upload.appendTo('#ej2_uploadBox')</pre>
Remove URL	Property: removeUrl <pre>\$('#uploadBox').ejUploadbox(   ({     removeUrl: 'removeUrl'   }) )</pre>	Property: asyncSettings.removeUrl <pre>var uploader =new ej.inputs.Uploader({   asyncSettings: { removeUrl: 'removeUrl' } })</pre>
Automaticall y upload the file when files added in to upload queue	Property: autoUpload <pre>\$('#uploadBox').ejUploadbox(   ({     autoUpload: false   }) )</pre>	Property: autoUpload <pre>var uploader =new ej.inputs.Uploader({   autoUpload: false })</pre>
Synchronous upload	Property: asynUpload <pre>\$('#uploadBox').ejUploadbox(   ({     ayncUpload: false   }) )</pre>	No Separate property for enabling synchronous upload.It can be enabling by using below configuration <pre>var uploader =new ej.inputs.Uploader({   autoUpload: false })</pre>
Key to get the selected files in server side	Property: uploadName <pre>\$('#uploadBox').ejUploadbox(   ({     saveUrl: 'saveUrl',     uploadName: 'UploadKey'   }) )</pre>	Id of the element used as key value

Upload the files dynamically	Not Applicable	Method: upload()  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: ""} }) uploader.appendTo('#ej2_Uploader')uploader.upload(filesData)
Event triggers before start to upload the action	Event: beforeSend  \$('#uploader').ejUploadbox({ beforeSend: function () {} })	Event: uploading  var uploader =new ej.inputs.Uploader({ uploading: beforeUploadStart }) uploader.appendTo('#ej2_Uploader')function beforeUploadStart() {}
Event triggers when the upload is in progress	Event: inProgress  \$('#uploader').ejUploadbox({ inProgress: function () {} })	Event: progress  var uploader =new ej.inputs.Uploader({ progress: uploadingInProgress }) uploader.appendTo('#ej2_Uploader')function uploadingInProgress () {}
Event triggers when upload got success	Event: success  \$('#uploader').ejUploadbox({ success: function () {} })	Event: success  var uploader =new ej.inputs.Uploader({ success: uploadingSuccess }) uploader.appendTo('#ej2_Uploader')function uploadingSuccess () {}
Event triggers when upload got failed	Event: error  \$('#uploader').ejUploadbox({ error: function () {} })	Event: failure  var uploader =new ej.inputs.Uploader({ failure: uploadingFails }) uploader.appendTo('#ej2_Uploader')function uploadingFails () {}
Event triggers when the upload got started	Event: begin  \$('#uploader').ejUploadbox({ begin: function () {} })	Not Applicable
Event triggers	Event: cancel  \$('#uploader').ejUploadbox({	Event: canceling  var uploader =new ej.inputs.Uploader({ canceling: uploadingCancel

when cancel the upload	cancel : function () {} })	}) uploader.appendTo('#ej2_Uploader')function uploadingCancel () {}
Event triggers when the upload completed	Event:complete \$('#uploader').ejUploadbox({ complete : function () {} })	Not Applicable

### Chunk upload

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enabling the chunk upload	Not Applicable	Property: asyncSettings.chunkSize  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000} }) upload.appendTo('#ej2_uploadBox')
Retry the upload automatically when it's get failed	Not Applicable	Property: asyncSettings.retryCount, asyncSettings.retryAfterDelay  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000, retryCount: 3, retryAfterDelay: 1000} }) upload.appendTo('#ej2_uploadBox')
Pause the uploading file	Not Applicable	Method: pause  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000 } }) upload.appendTo('#ej2_uploadBox') upload.pause(filesData)
Event triggers when pausing the file	Not Applicable	Event: pausing  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000 }, pausing: onFilePause })

		upload.appendTo('#ej2_uploadBox') function onFilePause() {}
Resuming the paused file	Not Applicable	Method: resume  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000, } }) upload.appendTo('#ej2_uploadBox') upload.resume(filesData)
Event triggers when resuming the file	Not Applicable	Event: resuming  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000, }, resuming: onFileResume }) upload.appendTo('#ej2_uploadBox') function onFileResume () {}
Retry the failed file	Not Applicable	Method: retry  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000, } }) upload.appendTo('#ej2_uploadBox') upload.retry(filesData)
Cancel the failed file	Not Applicable	Method: cancel  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000, } }) upload.appendTo('#ej2_uploadBox') upload.cancel(filesData)
Event triggers when cancel the file	Not Applicable	Event: canceling  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000 }, canceling: onFileCancel }) upload.appendTo('#ej2_uploadBox') function onFileCancel(){}

Event triggers when chunk file fails	Not Applicable	Event:chunkFailure  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000 }, chunkFailure: onChunkFail }) upload.appendTo('#ej2_uploadBox') function onChunkFail() {}
Event triggers when chunk file success	Not Applicable	Event: chunkSuccess  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', chunkSize: 50000 }, chunkSuccess: onChunkSuccess }) upload.appendTo('#ej2_uploadBox') function onChunkSuccess () {}

#### Remove action

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Remove the uploaded file	Not Applicable	Method: remove  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url', removeUrl:'removeUrl' } }) upload.appendTo('#ej2_uploadBox') upload.remove(filesData)
Event triggers when the file removing succeed	Event:remove  \$('#uploadBox').ejUploadbox({ remove: function() {} })	Event:success  var uploader =new ej.inputs.Uploader({ asyncSettings: { saveUrl: 'save url' }, Success: onRemoveSuccess }) upload.appendTo('#ej2_uploadBox') function onRemoveSuccess () {}
Event triggers when the file removing fails	Not Applicable	Event: failure var uploader =new ej.inputs.Uploader({

		<pre> asyncSettings: { saveUrl: 'save url', removeUrl: 'remove url' }, failure: onRemoveFailure }) upload.appendTo('#ej2_uploadBox') function onRemoveFailure () {} </pre>
--	--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Buttons

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Customize button text	Property: buttonText  <pre> \$('#uploadBox').ejUploadbox({ remove: function() {} }) </pre>	Property: buttons  <pre> var uploader =new ej.inputs.Uploader({ buttons: { browse: 'Choose File', clear: 'Clear files', upload: 'upload Files' } }) </pre>

## Drag and drop

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable drag and drop upload	Property: allowDragAndDrop  <pre> \$('#uploadBox').ejUploadbox({ allowDragAndDrop: true }) </pre>	No separate property to disabling drag and drop
Set custom drop area	Not Applicable	Property: dropArea  <pre> var uploader =new ej.inputs.Uploader({ dropArea: dropElement }) </pre>

## Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Adding custom class to wrapper element	Property: cssClass  <pre> \$('#uploadBox').ejUploadbox({ </pre>	Not Applicable

	cssClass: 'Custom-Class' })	
Enable/Disable the control	Property: enabled  \$('#uploadBox').ejUploadbox({ enabled: false }) Method: enable(), disable()	Property: enabled  var uploader =new ej.inputs.Uploader({ enabled: false })
Set height for uploader	Property:height  \$('#uploadBox').ejUploadbox({ height: '100%' })	Not Applicable
Set width for uploader	Property:width  \$('#uploadBox').ejUploadbox({ width: '100%' })	Not Applicable
Adding HTML attributes	Property:htmlAttributes  \$('#uploadBox').ejUploadbox({ htmlAttributes: {'aria-label': 'UploadBox'} })	Not Applicable
Event triggers when control created successfully	Event:create  \$('#uploadBox').ejUploadbox({ create: function () {} })	Event: created  var uploader =new ej.inputs.Uploader({ created: function() {} })
Event triggers when destroy the control	Event: destroy  \$('#uploadBox').ejUploadbox({ destroy: function () {} })	Not Applicable
Keeping the model values in cookies	Property: enablePersistence  \$('#uploadBox').ejUploadbox({ enablePersistence: true })	Property: enablePersistence  var uploader =new ej.inputs.Uploader({ enablePersistence: true })
Get the selected files data	Not Applicable	Method: getFilesData



		<pre>var uploader =new ej.inputs.Uploader() uploader.appendTo('#uploadbox') uploader.getFilesData()</pre>
Convert bytes in to MB, GB	Not Applicable	<p>Method: bytesToSize</p> <pre>var uploader =new ej.inputs.Uploader() uploader.appendTo('#uploadbox') uploader.bytesToSize(5000)</pre>